Chapter 15

Packet Loss

In a packet-switched network that does not provide mechanisms for reserving resources within the network on behalf of a particular packet "flow", loss is inevitable under conditions of load. The Internet is such a network. According to traditional network traffic theory, based on Poisson models that emphasize at most fleeting correlations between packet arrivals, one can generally engineer a packet-switched network to have as low a packet loss rate as desired. Operational experience, however, has been quite contrary and brutal to the Poisson framework [JR86, G90, FL91, DJCME92, PF95], which appears woefully inadequate for accurately predicting actual network behavior. Recent years have seen the rise of *self-similar* traffic models, in which correlations are extremely long-lived and have a fractal structure, leading to "burstiness on all time scales" [LTWW94]. Fractal models predict that packet loss is extremely hard to avoid, due to the great burstiness of network traffic, and, more generally, due to the lack of a single *burst time scale* for which one can then engineer the network to accommodate.

We should note that packet loss is not unequivocally a problem. TCP makes splendid use of packet loss as an *implicit* signal that the network is under stress and the TCP sender should reduce its sending rate [Ja88]. If the network had immense buffering within it to avoid packet loss, this over-engineering would defeat TCP's congestion signal. Furthermore, such buffering does *not* guarantee that the network can promise to always deliver useful throughput [Na87], and, actually, things would be worse off, since TCP senders then could not adapt their transmission rates to the limited capacity of the bottleneck link.

In this chapter we look at what our measurements tell us about packet loss in the Internet: how frequently it occurs and with what general patterns (§ 15.1); differences between loss rates of data packets and acks (§ 15.2); the degree to which it occurs in bursts (§ 15.3); the degree to which losses occur at the bottleneck link (§ 15.4); how loss rates evolve over time (§ 15.5); and how well TCP retransmission matches genuine loss (§ 15.6).

15.1 Loss rates

A fundamental issue in measuring packet loss is to avoid confusing measurement drops with genuine losses. Doing so can often be difficult unless the measurement apparatus takes pains to accurately report measurement drops. As we saw in § 10.3.1, some do and some do not. Here is one of the analysis areas where the effort to ensure that `tcpanaly` understands the details of the many

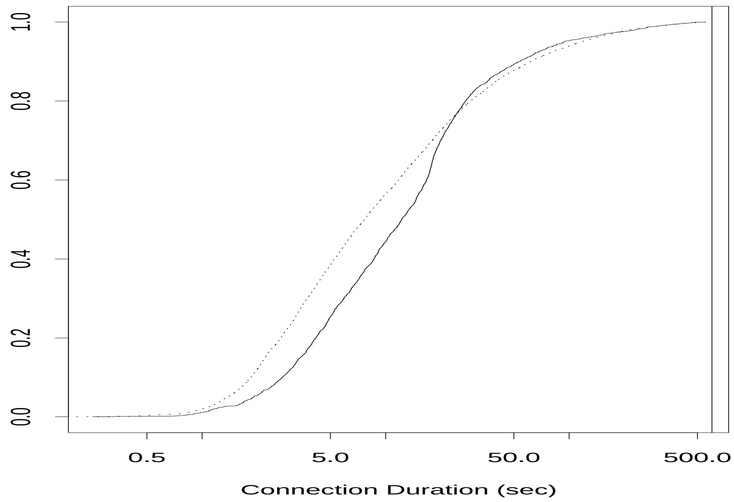

Figure 15.1: Connection durations for $\mathcal{N}_1$ (solid) and $\mathcal{N}_2$ (dotted)

TCP implementations in our study pays off extremely well. Because we can determine whether traces suffer from measurement drops, we can exclude those that do from our packet loss analysis and avoid what could otherwise be significant inaccuracies. Since, for the most part, measurement drops will be uncorrelated with the presence of true network drops, excluding these tainted traces should not bias our subsequent analysis. An exception would be if the measurement drops are due to large bursts of traffic on the local network overrunning the packet filter's ability to record the burst, and if such bursts were coupled with true loss on the local network. Since our interest lies in loss in the Internet-in-the-large, and not in loss in local networks (even though local loss also contributes to the end-to-end chain), we regard this source of bias as minor.

Our measurements do, however, suffer from one form of bias: due to their limited duration (§ 9.3), we will fail to successfully measure and analyze connections that suffered such high packet loss rates that they required more than 10 minutes to transfer 100 Kbyte. When these measurement attempts reach the 10-minute lifetime without having successfully completed, the entire measurement attempt is aborted, and *no* trace data is retrieved from the NPDs conducting the measurement.

Unfortunately, due to the centralized control of the experiment, we cannot accurately assess how often a measurement failed for this reason, and how often for a different reason, such as a loss of connectivity between `npd_control` and one of the remote NPDs (§ 5.2, § 9.3). Thus, the statistics presented in this section will *underestimate* Internet packet loss rates somewhat.

We argue, however, that the bias is, overall, fairly small. Figure 15.1 shows the distribution of the connection durations for $\mathcal{N}_1$ (solid line) and $\mathcal{N}_2$ (dotted line). The vertical line on the righthand side of the plot marks the 10-minute maximum duration. The x-axis is logarithmically scales, so we see that a large number of the connections in our study completed much sooner than the 10 minute upper lifetime. This in turn suggests that the lifetime was generally not a limitation. At the end of this section, however, we show that it *did* significantly bias European loss rates towards underestimation.

We begin our analysis with looking at aggregate packet loss over the course of entire connections. In $\mathcal{N}_1$, out of about 714 thousand packets (data and ack) transmitted, 3.0% failed to arrive at the other end. In $\mathcal{N}_2$, for 4.66 million packets, the figure rose to 4.6%, a significant increase that merits further investigation.

On immediate question is whether the use of additional sites in $\mathcal{N}_2$ (and the absence of a few of the $\mathcal{N}_1$ sites) skewed these basic numbers. Indeed, it did, but *towards underestimating the increase*! Of the sites in common, in $\mathcal{N}_1$, 2.7% of the packets were lost, while in $\mathcal{N}_2$, this figure nearly doubled to 5.2%. Conventional wisdom among TCP researchers holds that a loss rate of 5% has a significant adverse effect on TCP performance, because it will greatly limit the size of the congestion window and hence the transfer rate, while 3% is often substantially less serious. Thus, it behooves us to try to understand the circumstances and details of the increase as much as possible.

First, we need to address the question of whether the increase in loss rate was due to the use of bigger windows in $\mathcal{N}_2$ than in $\mathcal{N}_1$ (§ 9.3). Such could easily be the case, since with larger windows the transfers will often have significantly more data in flight, and, consequently, will load the router queues along the path much more. We can assess the impact of larger windows by looking at loss rates of *data* packets versus those for *ack* packets. Data packets contribute to queueing and having more in flight stresses the forward path. On the other hand, the rate at which a TCP transmits data packets *adapts* to current conditions. Ack packets contribute almost no additional load along the reverse path, other than occupying a buffer when queued, so having more of them in flight at one time should not significantly alter the loss rate they suffer. They do not adapt to current conditions, except during periods of heavy congestion, when an entire window's worth of acks is lost, forcing a timeout retransmission.[1] Thus, to compare changes in loss rates between $\mathcal{N}_1$ and $\mathcal{N}_2$, using the ack loss rates should eliminate the bias caused by the different window sizes. We discuss more issues concerning data packet loss versus ack loss in § 15.2.

Overall, in $\mathcal{N}_1$, acks were actually slightly more likely (3.16%) to be lost than data packets (2.96%), while in $\mathcal{N}_2$ the ordering is the opposite (4.25% for acks versus 4.75% for data packets). Restricting the comparison to the sites in common, however, changed the discrepancy between data packets and acks, with 2.88% for acks versus 2.65% for data packets in $\mathcal{N}_1$, and 5.14% versus 5.28% for the same $\mathcal{N}_2$ figures. So, even if we restrict ourselves to the ack loss rates for the common sites, which should be quite sound to compare, we observe a 78% increase in the loss rate, from 2.88% to 5.14%.

Another interesting loss rate figure is how the rate changes if we condition on observing at least one loss during the connection. Here we make a tacit assumption that a network path has two basic states, "quiescent," during which connections tend to not suffer any loss, and "busy," during which they tend to suffer loss. The first corresponds to, overall, light or steady enough load that the router buffers suffice to avoid packet loss, and the second to sufficient load, overall, to occasionally overflow the buffers. We would expect to find that "busy" states coincide with the usual peak usage times of working hours, and quiescent states with off-peak times. We return to this point below, in the discussion of Figure 15.3 and Figure 15.4.

In $\mathcal{N}_1$, 52% of the connections between the common sites did not lose a single ack packet. However, only 28% of the connections losing at least one ack lost exactly one. For $\mathcal{N}_2$, the corre-

[1] The transmission rate of acks can also adapt to current conditions if the loss conditions along both directions of the path are correlated, since the rate at which a TCP transmits acks reflects the rate at which it receives data packets. In § 15.2 below, however, we find that loss rates in the two directions are nearly uncorrelated.

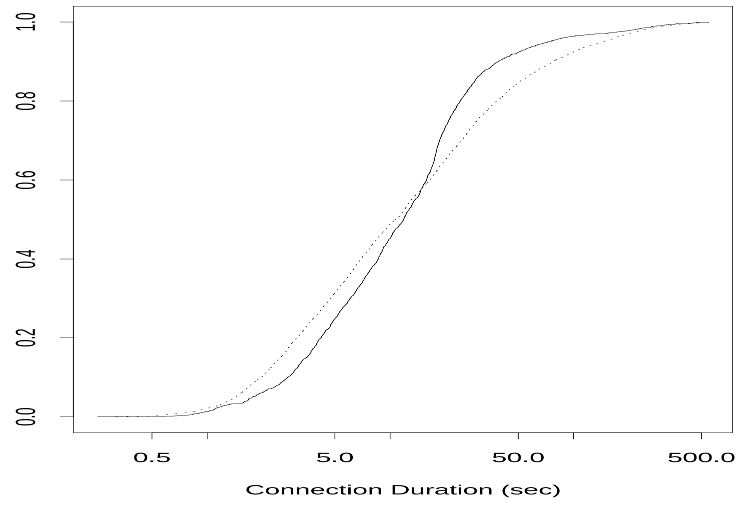

Figure 15.2: Connection durations for sites common to $\mathcal{N}_1$ (solid) and $\mathcal{N}_2$ (dotted)

sponding figures are 49% and 20%. We see that part of the change in the higher $\mathcal{N}_2$ ack loss rates stems from *greater loss during busy periods*. The proportion of quiescent periods remains virtually unchanged. Similarly, for the common sites, if we condition on a connection suffering at least one loss, then the ack loss rate for an $\mathcal{N}_1$ connection climbs from 2.88% to 5.69%, while for $\mathcal{N}_2$ the increase goes from 5.14% to 9.16%. Thus, even in $\mathcal{N}_1$, if the network path was busy (using our simplistic definition above), loss rates were quite high, and for $\mathcal{N}_2$ they shot upward to a level that in general will seriously impede TCP performance.

These increases give us strong evidence that networking conditions in one important respect *degraded* during the course of 1995, similar to our earlier finding that several aspects of Internet routing degraded during 1995 (§ 6.10, § 8.5). Since bottleneck link rates generally *increased* during 1995 (§ 14.7.1), we cannot tell from just the loss rate statistic whether users perceived the network as delivering better or worse service. A basic measure of perceived level of service is how long it takes to transfer a given amount of data. However, when comparing such durations we need to keep in mind that the use of bigger windows in $\mathcal{N}_2$ gave $\mathcal{N}_2$ connections more opportunity both to "fill the pipe" and to utilize fast retransmission (§ 9.2.7), which gives them performance advantages that have little to do with how the network service changed. (For the sites in common, in $\mathcal{N}_1$ the mean number of fast retransmissions was 0.98, while in $\mathcal{N}_2$ it climbed to 1.64.)

Still, we find the comparison illuminating. Figure 15.2 shows the distribution of the durations of connections between sites common to both $\mathcal{N}_1$ (solid line) and $\mathcal{N}_2$ (dotted line). For the sites in common, the median connection duration diminished from 11.8 sec in $\mathcal{N}_1$ to 10.7 sec in $\mathcal{N}_2$, a rather modest improvement. That single figure does not tell the entire story, though, since we see from the figure that the distribution of durations did not unilaterally slide a bit to the left. Instead, $\mathcal{N}_2$ connections were likely to be 20% shorter than those in $\mathcal{N}_1$ *if they were short*, meaning that we condition on the duration being < 12 sec; and 50% longer if we condition on the duration being > 12 sec. It seems likely that the differences are due to a higher prevalence of fast retransmission

Region	# $\mathcal{N}_1$	# $\mathcal{N}_2$	$\mathcal{N}_1$ loss rate	$\mathcal{N}_2$ loss rate	Δ
Europe	104	734	2.8%	2.8%	−03%
North America	641	2,405	1.3%	1.6%	+23%
(`umont`)	75	562	1.5%	5.8%	+287%
Into Europe	255	1,243	6.2%	11.7%	+88%
Into North America	320	1,544	3.5%	3.2%	−08%
All regions	1,395	6,488	2.8%	4.6%	+63%

Table XXI: Ack loss rates for different connection geographies

in $\mathcal{N}_2$ aiding short transfers, while a higher packet loss rate led to more frequent timeouts for those connections that failed to open their congestion windows enough to facilitate fast retransmission.[2]

So far, we have treated the Internet as a single aggregated network in our loss analysis. Geography, however, plays a crucial role in the prevalence of packet loss. To study geographic effects, we partition the connections between the sites common to $\mathcal{N}_1$ and $\mathcal{N}_2$ into four primary groups: "European," "North American," "Into Europe," and "Into North America." European connections are those with both a European sender and a European receiver. North American have both sender and receiver in Canada or the United States (but see below). "Into Europe" are connections with European data *senders* and North American data *receivers*. The terminology is backwards here because what we will assess are *ack* loss rates, and these are generated by the receiver. Hence, "Into Europe" loss rates reflect those experienced by packet streams traveling from North America into Europe. Similarly, "Into North America" are connections with North American data senders, European data receivers, and ack streams traveling from Europe into North America.

This partition does not include connections to or from Australia, because we had only one Australian site common to both $\mathcal{N}_1$ and $\mathcal{N}_2$, so it would be difficult to gauge the generality of loss rates involving it. We note, however, that it experienced a rise of more than a factor of two in the loss rates of ack traveling into and out of Australia, from 3.3% in $\mathcal{N}_1$ to 7.8% in $\mathcal{N}_2$.

While the above grouping was our original intent, upon examining the data we made one further distinction. The sole Canadian site, `umont`, was a major outlier for packet loss in $\mathcal{N}_2$, so large that its presence as one of the 13 North American hosts sufficed to significantly skew the overall North American findings. (It was not, however, an outlier in $\mathcal{N}_1$.) Since we had no other Canadian sites in our study, we cannot gauge whether this reflects a problem unique to `umont` or a more general problem with Canadian Internet service. Consequently, we removed `umont` from our notion of "North America" as described above; so, in fact, all of the North American sites discussed below are in the United States. We also summarize below connections from U.S. sites to or from `umont`, to illustrate its atypical loss rates.

Table XXI shows the loss rates of ack packets for the different regions. The second and third columns give the number of $\mathcal{N}_1$ and $\mathcal{N}_2$ connections that occurred in the region. There were 6 common European sites and 12 North American sites plus `umont`. The fourth and fifth columns give the overall loss rate for the ack packets sent during all of the region's connections, and the final column indicates the loss rate change between $\mathcal{N}_1$ and $\mathcal{N}_2$. Clearly:

[2]Note that, had we not restricted ourselves to the sites common to the two datasets, but instead interpreted Figure 15.1 in this regard, then we would have drawn a considerably different, less sound conclusion.

Region	$\mathcal{N}_1$ quies.	$\mathcal{N}_2$ quies.	$\mathcal{N}_1$ cond. loss	$\mathcal{N}_2$ cond. loss	Δ
Europe	48%	58%	5.3%	5.9%	+11%
North America	66%	69%	3.6%	4.4%	+21%
(`umont`)	60%	15%	3.7%	6.8%	+81%
Into Europe	40%	31%	9.8%	16.9%	+73%
Into N.A.	35%	52%	4.9%	6.0%	+22%
All regions	53%	52%	5.6%	8.7%	+54%

Table XXII: Conditional ack loss rates for different connection geographies

- Europe suffered considerably higher packet loss rates than did North America, but the loss rate appears stable. However, we show below that the European figures are *biased* towards *underestimation*;
- North American loss rates were fairly low and, while the trend is increasing, it is not doing so at an ominous rate;
- `umont` suffered a tremendous increase in packet loss rate, although we lack sufficient data to tell if this is a general problem for Canadian networks or specific to the University of Montreal or its local region;
- the trans-Atlantic links carrying European traffic to North America had fairly high loss rates, but the situation is perhaps improving; and
- the links carrying North American traffic to Europe were a compounding disaster. We note that since Europe's rates are significantly lower than those of trans-Atlantic traffic heading into Europe, it must be the case that most traffic between two European sites stays inside Europe, rather than transiting through North America, even though we sometimes observed such routes in § 6.9.

Table XXII looks at loss rates for the same regions, but now with conditioning on whether any acks were lost. The second and third columns give the proportion of quiescent connections, where "quiescent" is defined as above to mean connections that did not lose any acks. We see that, except for `umont` and the trans-Atlantic links going into North America, the proportion of quiescent connections was fairly stable, suggesting that perhaps changes in loss rate are confined to already-loaded "busy" periods of heavy load. We investigate this possibility in more detail shortly.

The fourth and fifth columns list the proportion of acks lost, given that at least one ack was lost, and the final column summarizes the relative change. None of the conditional loss rates is especially heartening, and the trends are *all* increasing. During $\mathcal{N}_2$, the trans-Atlantic links into Europe were close to unusable during busy periods, with a loss rate of nearly 17%. This matches anecdotal reports such as requests the author received to mail hardcopies of papers to European researchers since they could not viably retrieve them over the network. In summary, we note that, *for every region, loss rates for busy connections increased between* $\mathcal{N}_1$ *and* $\mathcal{N}_2$.

Within regions, we find considerable site-to-site variation in loss rates, as well as variation between loss rates for packets inbound to the site and those outbound (§ 15.2). We did not, however,

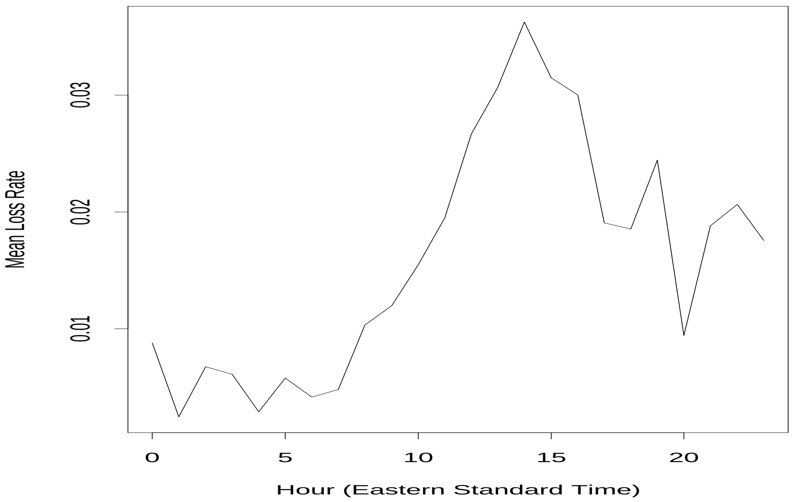

Figure 15.3: Hourly variation in ack loss rate for North American connections

find any other outliers as dramatic as `umont` in $\mathcal{N}_2$, so we kept the regions otherwise intact.

The last aggregate loss statistics we look at are variations of loss rate over the course of the day. We expect to find a diurnal cycle, as numerous studies have noted significant hourly variation in connection and packet arrival rates ([PF95] and many others). It was this expectation that led us to postulate that the distinction made above between "busy" and "quiescent" connections is broadly meaningful.

Figures 15.3 and 15.4 show the hourly loss rates for the $\mathcal{N}_2$ connections internal to North America and Europe, respectively. The North American loss rates, with the x-axis reflecting the hour in the Eastern Standard Time zone, clearly follow the oft-observed pattern of activity increasing over the morning hours and falling off during the late afternoon. [PF95] notes a pickup in evening FTP traffic, which agrees with the secondary peak. One unusual facet of Figure 15.3 is that it does not exhibit a noon-time "dip." However, this is almost certainly due to the North American traffic spanning three time zones, effectively spreading out lunch-related lulls over several hours. The apparent discontinuity between the 23rd hour at the right and the midnight hour at the left, however, is puzzling. We have verified that as one approaches midnight, the rates come closer together. We do not, though, have an explanation as to why midnight EST would serve as such a sharp transition point, given that it corresponds to 9PM Pacific Standard Time, when presumably we still see considerable user activity.

Figure 15.4 differs considerably from Figure 15.3. Here the x-axis reflects the hour in the Greenwich Mean Time zone. We observe a morning rise in loss rate, but a considerable noontime dip lasting several hours, followed by a striking increase in the late afternoon. Again, the evening hours are elevated compared to the early morning hours, with a sharp transition occurring around midnight. The late afternoon hours may in part reflect increasing background traffic from North American sites, too, since late afternoon GMT coincides with noon and early afternoon EST, which we see in Figure 15.3 is the peak North American period.

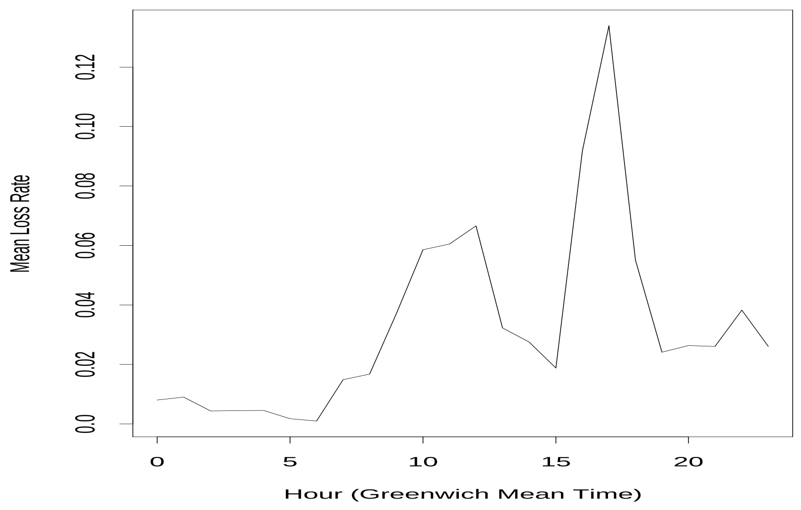

Figure 15.4: Hourly variation in ack loss rate for European connections

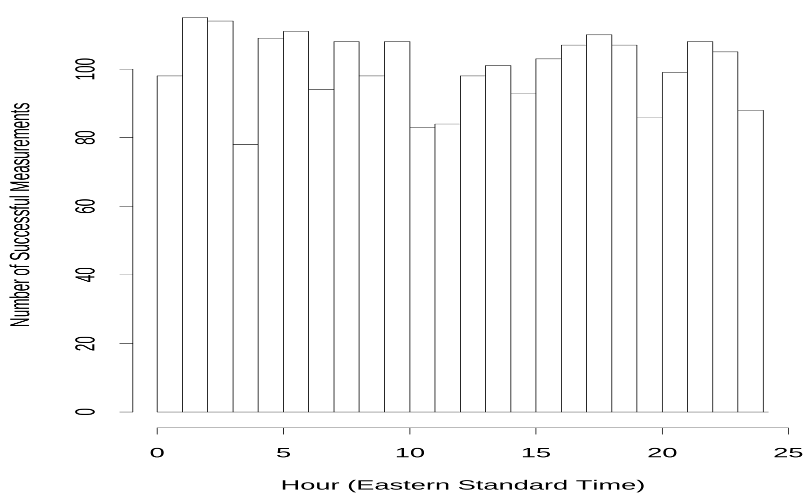

Figure 15.5: Successful North American measurements, per hour

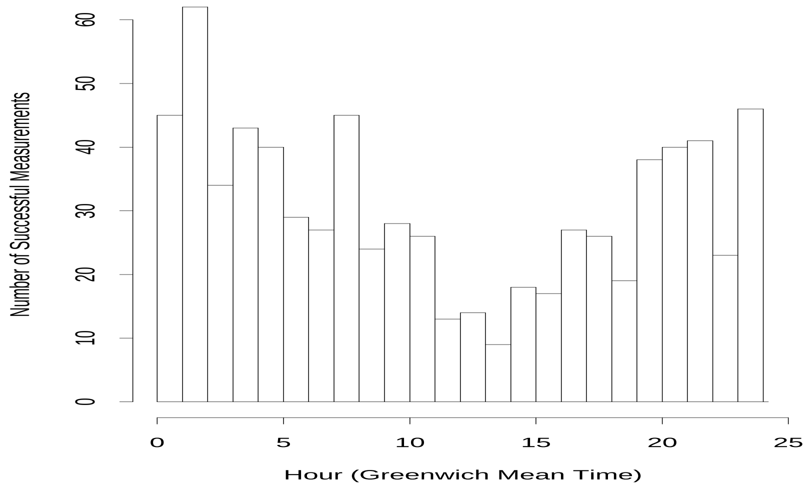

Figure 15.6: Successful European measurements, per hour

We must exercise caution, however, in interpreting Figure 15.4, due to our measurement bias against very long-lived connections (discussed at the beginning of this section). We can test for the presence of this bias by examining how many successful measurements we made for each hour of the day. Because of our Poisson sampling methodology (§ 9.1), measurement *attempts* were uniformly distributed over the course of the day. Figure 15.5 shows a histogram of the number of successful North American measurements made for each distinct hour of the day. The distribution appears fairly even, and, indeed, the measurement times pass the powerful Anderson-Darling A^2 goodness-of-fit test for uniformity [DS86], using 5% significance (and, indeed, for higher significance).

Figure 15.6 shows the same histogram for the European measurements. The bias towards the less busy early morning and late evening hours immediately stands out. The distribution fails A^2 at all significance levels, as one might expect. The bias is strongest against the 11AM to 1PM periods, and eases somewhat in the later afternoon, so the apparent difference between the two corresponding peaks in Figure 15.4 may be simply due to measurement bias and not reflect a true underlying difference. However, we can certainly conclude based on Figure 15.6 that our analyses of European loss rates are in general *underestimates*.

15.2 Data packet loss vs. ack loss

We noted in the previous section that analyzing data packet loss rates can be complicated because the size of the data packets and the tendency for them to be sent closely together both add to queueing load along the network path. We expect that this load in turn leads to a greater likelihood of the data packets being lost, though, because TCP can unfairly distribute available bandwidth [FJ92], this is not necessarily the case. We saw in § 15.1 that, in $\mathcal{N}_1$, acks were actually slightly more likely to be lost than data packets, though, in $\mathcal{N}_2$, the pattern reverses, which we (at least partially) attribute to the use of bigger windows in $\mathcal{N}_2$ (§ 9.3).

In this section we take a closer look at the loss rates of data packets versus those of acks. We consider any packet carrying one or more bytes of user data as a data packet. We would expect to observe some differences between different-sized data packets. Unfortunately, it would prove difficult to explore this effect with our data. Some of the sites in our study always used a maximum segment size (MSS) of 512 bytes, the common default value, while others used larger sizes whenever the opportunity to do so arose. But the site-specific nature of the MSS used means that, for each site, the samples of data packet loss rates generally reflect only a small number of packet sizes, sometimes only one. Since in § 15.1 we showed that ack loss rates exhibit strong regional variation, we could easily conflate a spurious MSS size effect in data loss rates with a genuine, separate effect due to the regions.

Thus, we confine ourselves to a simple definition of "data packet" as one carrying any user data whatsoever. But in addition, we make a key distinction between "loaded" and "unloaded" data packets. A "loaded" data packet is one that presumably queued at the bottleneck link behind one of the connection's previous packets, while an unloaded data packet is one that we know did not *have* to queue at the bottleneck behind a predecessor. Here we are abstracting the intricate, multi-element network path to a presumably equivalent model of a single element that forwards at the bottleneck rate, and at which all significant queueing occurs.

To tell if a packet is unloaded, we first form an estimate of the bottleneck bandwidth using the methodology developed in Chapter 14. If the methodology indicates a bottleneck change or the possible presence of a multi-channel bottleneck, then we refrain from further packet-loss analysis.

If, however, the methodology produces a single bottleneck estimate, ρ_B, as is generally the case, then the methodology also associates lower and upper bounds with ρ_B (Eqn 14.12):

$$\rho_B^- < \rho_B < \rho_B^+. \tag{15.1}$$

This equation in turn gives us the maximum amount of time required for a b-byte packet to transit across the bottleneck, namely:

$$\tau_b^+ = b/\rho_B^- \ \text{sec}. \tag{15.2}$$

Let T_i^s be the time at which the sender transmits the ith data packet. We then sequentially associate a *maximum load* λ_i^+ with the packet as follows. The first packet has a load equal to

$$\lambda_1^+ = \tau_b^+, \tag{15.3}$$

where b is the size of the packet. Subsequent packets have a load

$$\lambda_i^+ = \tau_b^+ + \max\left[(T_{i-1}^s + \lambda_{i-1}^+) - T_i^s, 0\right]. \tag{15.4}$$

λ_i^+ thus reflects the maximum amount of extra delay the ith packet incurs due to its own transmission time across the bottleneck link, plus the time required to first transmit any preceding packets across the bottleneck link, if i will arrive at the bottleneck before they completed transmission. The latter will be the case if

$$T_i^s < T_{i-1}^s + \lambda_{i-1}^+, \tag{15.5}$$

because this condition means that packet i was sent shortly enough after packet $i-1$ that packet $i-1$ would not yet have cleared the bottleneck link by the time packet i arrived at the bottleneck.

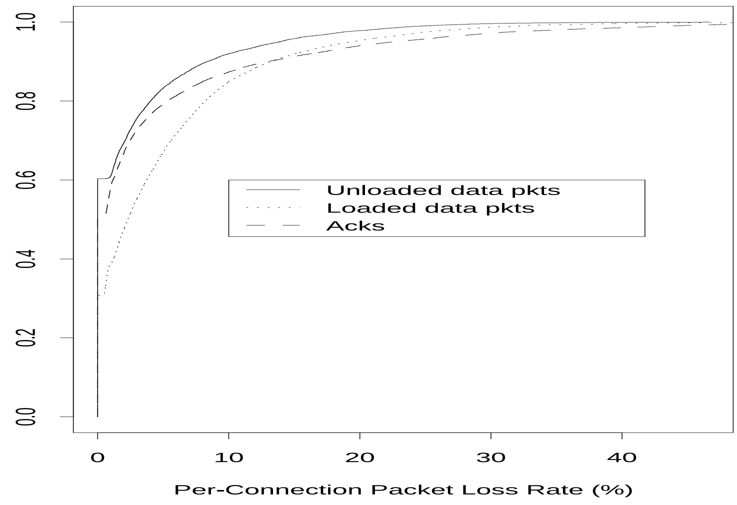

Figure 15.7: $\mathcal{N}_2$ loss rates for data packets and acks

If Eqn 15.5 applies to packet i, then we will say that packet i was "loaded," meaning that it had to wait for pending transmission of earlier packets. Otherwise, we term it "unloaded."

The development of the maximal load λ_i^+ has natural analogs λ_i^- and λ_i for the minimal and central loads associated with each packet, by using ρ_B^+ or ρ_B (from Eqn 15.1) in Eqn 15.2 to compute analogous "self-interference" time constants τ_b^- and τ_b.[3] Similarly, we can define different flavors of "loaded" and "unloaded" depending on whether we use the maximal, central, or minimal definitions for λ_i. In this section, we exercise conservatism and only consider a packet as unloaded for the definition in terms of the maximal λ_i^+.

Presently, our interest in whether a packet is loaded or unloaded comes just from analyzing whether the two types have different loss patterns. In § 16.2.6 we look in more detail at the coupling between λ_i and the variation in packet transit times.

In both $\mathcal{N}_1$ and $\mathcal{N}_2$, about 2/3's of the data packets were loaded. We might at first expect more loaded packets in $\mathcal{N}_2$, due to its use of bigger windows. Window size, however, determines whether the bottleneck link might continuously *remain* loaded. Even for a relatively small window size, the TCP sender will transmit a number of packets (equal to the window size) over a fairly short amount of time, and all of these but the first will be loaded. Once the entire window is in flight, then a lull comes equal to the mismatch between the small window and the bandwidth-delay product corresponding to the bottleneck rate and the RTT. Then the acks for the window arrive in short order, and self-clocking leads to another flight of all-but-one loaded packets. Thus, window size does not have a great deal of impact on the proportion of loaded packets.

Figure 15.7 shows the distributions of loss rates during $\mathcal{N}_2$ for unloaded data packets, loaded data packets, and acks. All three distributions show considerable probability of zero loss.[4]

[3] ρ_B^+ is associated with λ_i^- because of the inverse relationship given by Eqn 15.2; the higher the bottleneck bandwidth, the lower the time required for a packet to transit across the bottleneck, so the less load associated with the packet.

[4] Each curve also shows a horizontal shift just above a loss rate of 0%. These reflect the fact that the loss rate is

From the figure, we immediately see that loaded packets are much more likely to be dropped than unloaded packets, as we would expect. In addition, we see that acks are consistently more likely than unloaded packets to be dropped, but generally less likely to be dropped than loaded packets, except during times of severe loss, above about 14%, which make up the upper 10% of the distributions. We interpret the difference between ack and data loss rates as reflecting the fact that, while an ack stream presents a much lighter load to the network than a data packet stream (particularly a series of loaded data packets), the ack stream does *not* adapt to the current network conditions, while the data packet stream does. Thus, unloaded data packets gain the twin benefits of traveling at a time when the connection is not itself significantly contributing to load along the network path, and also lowering their transmission rate during times of congestion. Loaded data packets stress the network path, but at least they adapt, and, during periods of heavy congestion, their adaptive behavior outweighs the advantages of ack streams that otherwise favor acks during periods of lower congestion.

The equivalent set of distributions for $\mathcal{N}_1$ is qualitatively the same, though the distance between the three distributions is narrower. This likely reflects both the overall lower loss rates in $\mathcal{N}_1$ (§ 15.1) and the use of smaller windows limiting loss rates for loaded packets.

It is interesting to note the extremes that packet loss can reach. In $\mathcal{N}_2$, the largest unloaded data packet loss rate we observed was about 47%. For loaded packets it climbed to 65%. As we would expect, these connections suffered egregiously, achieving overall data throughput rates in the low hundreds of bytes per second due to lengthy, backed-off timeout periods. However, they *did* manage to successfully complete their transfers within their alloted ten minutes, a testimony to TCP's tenacity. For both of these extremes, *no* acks were lost in the reverse direction! The largest ack loss rate was even higher, 68%. Starved for confirmation of forward progress, this connection also managed only a few hundred bytes per second. Ironically, *no* data packets were lost in the forward direction!

As indicated by these extreme cases, clearly packet losses on the forward and reverse paths are sometimes completely independent. Indeed, the coefficient of correlation between combined (loaded and unloaded) data packet loss rates and ack loss rates in $\mathcal{N}_1$ was about 0.21, with the correlation for connections within North America falling to 0.13. In $\mathcal{N}_2$, however, the loss rates become uncorrelated (coefficient of -0.02), perhaps due to the greater prevalence of significant routing asymmetry (Chapter 8).

Another form of asymmetry is the degree to which loss correlates with the connection's throughput. We would expect that data packet loss rates correlate more strongly, and negatively, with throughput, since each loss requires a retransmission that subsequently cuts the sender's transmission rate, and perhaps entails a lengthy timeout lull. Ack loss, on the other hand, may go unnoticed, if light, since acks are cumulative, and, if another ack arrives shortly, the connection will not stall for any appreciable amount of time.

To fairly gauge the correlation, we need to first account for the different maximum throughput rates due to the different bottleneck bandwidth rates. We do so by dividing the achieved throughput over the entire connection (total bytes transferred divided by total duration) by the estimated bottleneck bandwidth. We then compute θ, the coefficient of correlation, between the *logarithm* of the normalized throughput and the loss rates of interest, where the logarithmic transforma-

computed in terms of k packets lost out of a total of n, hence $1/n$ is the minimum possible positive loss rate. Since, for most connections, $n \approx 200$ packets, we observe a minimum possible loss rate of around 0.5%.

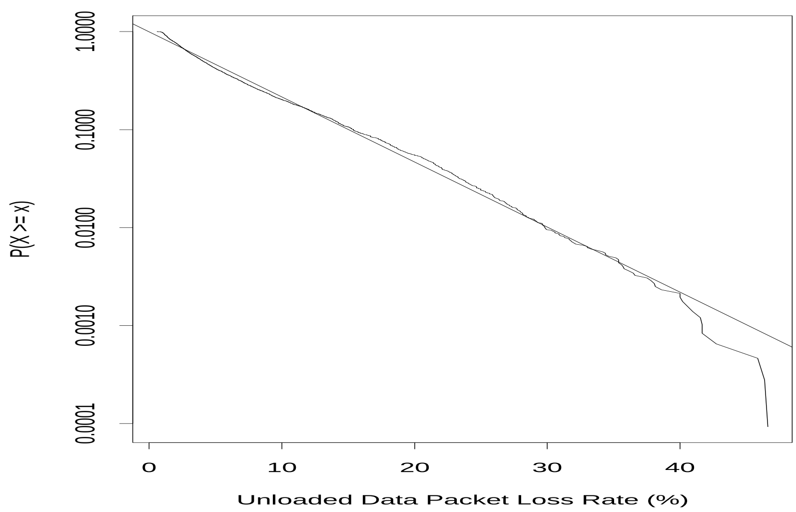

Figure 15.8: Complementary distribution plot of $\mathcal{N}_2$ unloaded data packet loss rate

tion is to reduce the otherwise dominating effect of throughput outliers.

For $\mathcal{N}_2$, we find that θ for the overall data loss rate is quite large, about -0.52, with unloaded loss rates a bit more strongly correlated than loaded loss rates. Presumably this latter effect is because backed-off timeout retransmissions, which have the greatest deleterious effect on connection throughput, always generate unloaded data packets, and further back-off occurs when these packets are then lost. The corresponding θ for ack loss rates also indicates a fairly strongly correlation, with a value of -0.42. Since these figures are for $\mathcal{N}_2$, this correlation is *not* due to any coupling between the ack loss rate and the data packet loss rate, because the two are generally uncorrelated, as shown previously. Instead, the correlation is probably due to the coupling between the ack loss rate and the possibility of losing an entire flight's worth of acks, which then unavoidably leads to a timeout retransmission (§ 15.6).

The significant correlation between ack loss rates and normalized connection throughput indicates that, when attempting to predict a connection's throughput along a particular forward path, it pays to have information about conditions along the reverse path, too. For the North American region (as defined in § 15.1), the correlations weaken somewhat, to -0.40 for data packet loss rates and -0.25 for acks. Thus, we must recognize that the strength of the correlations varies considerably.

The distributions in Figure 15.7 have shapes suggestive of exponential distributions, if we ignore the considerable zero portion of each distribution. Further investigating the distributions, one striking feature we find is that the non-zero portion of both the unloaded and loaded data packet loss rates is almost exactly exponential, while that for acks is not nearly so close a match.

Figures 15.8, 15.9, and 15.10 show logarithmically scaled complementary distribution plots of the unloaded, loaded, and ack loss rates, conditioned on observing at least one loss. A straight line on such a plot corresponds to an exponential distribution. We have added least-squares fits to each plot. We see that, for both unloaded and loaded data packets, the loss rate distribution is

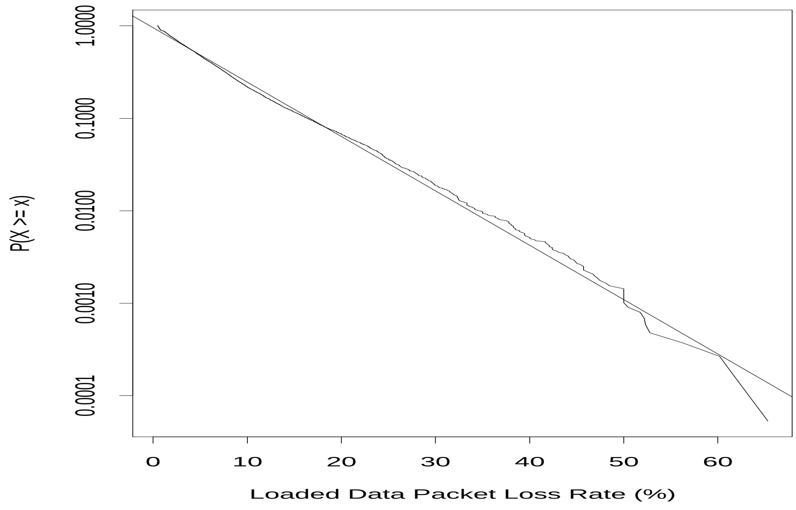

Figure 15.9: Complementary distribution plot of $\mathcal{N}_2$ loaded data packet loss rate

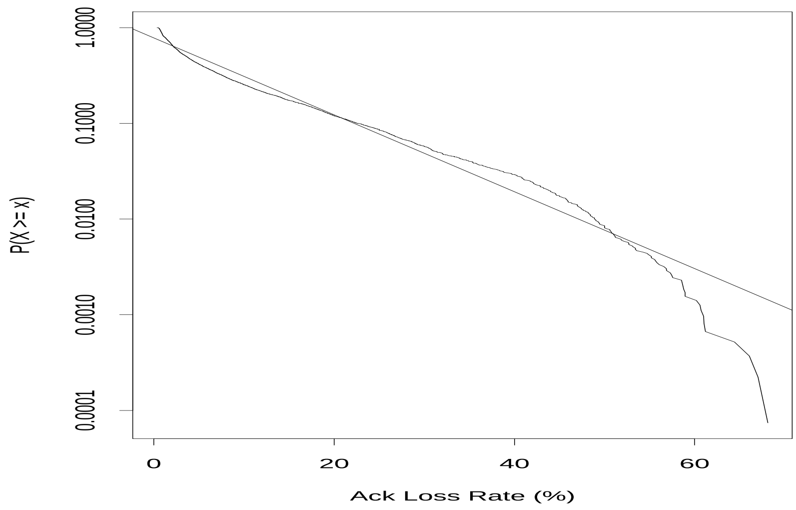

Figure 15.10: Complementary distribution plot of $\mathcal{N}_2$ ack loss rate

quite close to exponential, but for acks it deviates considerably more. The effect is widespread: it is also present for $\mathcal{N}_1$, and for the North American and European subsets of $\mathcal{N}_2$.

While striking, interpreting the fit to the exponential distribution is difficult. If, for example, packet loss occurs independently and with a constant probability, then we would expect the loss rate to reflect a binomial distribution, but that is *not* what we observe. (We also know from the results in § 15.1 that there is *not* a single Internet packet loss rate, or anything approaching such a situation.)

It seems likely that the better exponential fit for both loaded and unloaded data loss rates than ack loss rates holds a clue. The most salient difference between the transmission of data packets and that of acks is that the rate at which the sender transmits data packets *adapts* to the current network conditions, and furthermore it adapts *based on observing data packet loss*. Thus, if we passively *measure* the loss rate by observing the fate of a connection's TCP data packets, then we in fact are making measurements using a mechanism whose goal is to lower the value of what we are measuring (by spacing out the measurements). Consequently, we need to take care to distinguish between measuring overall Internet packet loss rates, which is best done using *non-adaptive* sampling, versus measuring loss rates *experienced* by a transport connection's packets—the two can be quite different.

15.3 Loss bursts

In this section we look at the degree to which packet loss occurs in *bursts* of more than one consecutive loss. Analytic models of network behavior often assume individual packet losses occur at a fixed rate but independently from other losses, as this assumption aids in keeping the models tractable. Accordingly, to gauge the strength of these models we need to address the issue of the soundness of this assumption.

As with loss rates, we expect that the size of loss bursts depends on whether we analyze losses of loaded data packets, unloaded data packets, or acks. These each correspond to a different transmission rate, and, furthermore, the first two are generated at a rate dynamically adapted to the frequency of previously observed packet loss, while acks are not.

The first question we address is the degree to which packet losses are well-modeled as independent. In [Bo93], Bolot investigated this question by comparing the unconditional loss probability, which we denote as P_l^u (*ulp* in Bolot's paper), with the conditional loss probability, P_l^c (*clp*), where P_l^c is conditioned on the fact that the previous packet was also lost. He found that $P_l^c \geq P_l^u$ always held, which one would expect, as it would be surprising if loss of the previous packet made loss of the next packet less likely. He investigated the relationship between P_l^u and P_l^c for different packet spacings δ, ranging from 8 msec to 500 msec. He found that P_l^c approaches P_l^u as δ increases, indicating that loss correlations are short-lived, and concluded that "losses of probe packets are essentially random as long as the probe traffic uses less than 10% of the available capacity of the connection over which the probes are sent." He also observed that P_l^u stabilized at about 10%, quite a high loss rate, though the path being studied included a heavily loaded trans-Atlantic link, and also a mid-level network known to have previously experienced 3% loss rates unrelated to congestion.

Table XXIII summarizes P_l^u and P_l^c for the different types of packets and our two datasets. P_l^c conditions on whether the connection's previous packet was lost, even if it is a different type than its successor (e.g., a loaded packet lost followed by an unloaded). Clearly, for TCP packets (which

Type of loss	P_l^u		P_l^c	
	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_1$	$\mathcal{N}_2$
Loaded data pkt	2.8%	4.5%	49%	50%
Unloaded data pkt	3.3%	5.3%	20%	25%
Ack	3.2%	4.3%	25%	31%

Table XXIII: Unconditional and conditional loss rates for different packet types

have a large range of interarrival intervals), we must discard the assumption that loss events are well-modeled as independent. Even for the low-burden, relatively low-rate ack packets, the loss probability jumps by a factor of seven if the previous ack was lost. We would expect to find the disparity strongest for loaded data packets, as these must contend for buffers with the connection's own previous packets, as well as any additional traffic, and indeed this is the case. We find the effect least strong for unloaded data packets, which accords with these not having to contend with the connection's previous packets.

It is interesting to observe that loaded packets are unconditionally less likely to be lost than unloaded packets. We suspect this reflects the fact that lengthy periods of heavy loss or outages will lead to timeout retransmissions, and these are unloaded, so they contribute to the loss probability of unloaded packets rather than loaded packets.

The relative differences between P_l^u and P_l^c in Table XXIII all exceed those computed by Bolot by a large factor. His greatest observed ratio of P_l^c to P_l^u was about 2.5:1. However, his P_l^u's were all much higher than those in Table XXIII, even for $\delta = 500$ msec, suggesting that the path he measured differed considerably from a "typical" path in our study.

(We also note that, since TCP packet loss events are not well-modeled as independent, it behooves us in general to avoid discussing unconditional packet loss in terms of *probability*, since for networking analysis this stochastic term often carries with it an implicit assumption of independence among the events. We advocate instead consistent use of the term packet loss *rate*, since this term downplays the implication of independence.)

Given that packet losses occur in bursts, the next natural question is: how big? To address this question, we grouped successive packet losses into *outages* and computed for each outage the number of packets lost and the duration of the outage in terms of the difference between the sending times of the two successfully arriving packets delimiting the outage. (Note that a data packet outage can encompass both loaded and unloaded packets.)

Figure 15.11 shows the distributions of the outage durations for data packets and acks in $\mathcal{N}_1$ and $\mathcal{N}_2$, using a logarithmic x-axis. We see considerable variation for the length of small outage durations. Our definition of duration as the time between two successfully arriving packets spanning the outage means the durations are both *upper bounds*,[5] and hence will be considerably skewed, for small values, by variations in the inter-packet spacing. The distributions are really only solid for larger values. Above 200 msec, the distributions agree quite closely, except that $\mathcal{N}_2$ data packet

[5]However, for large estimates the degree of overestimation is limited by the retransmit timer backoff (§ 9.2.3), and hence the estimated duration is off by at most a factor of two. Since we analyze the distributions using logarithmic x-axes, this factor at most results in a translation of the distribution's body—it does not appreciably alter the shape of the log-transformed distribution.

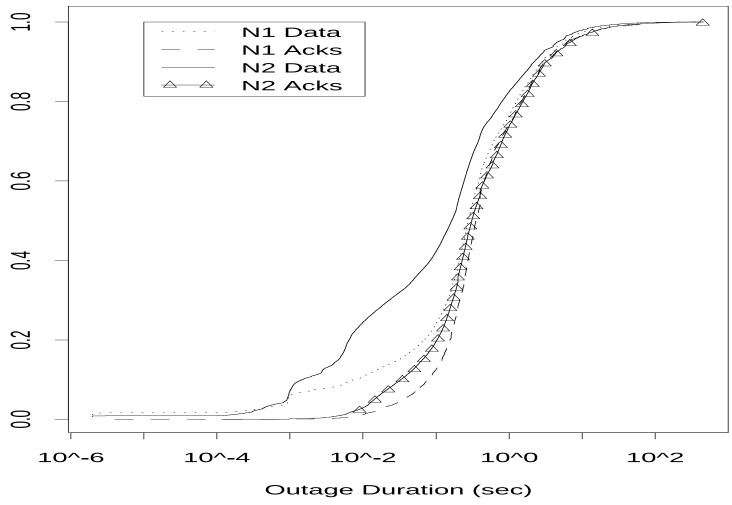

Figure 15.11: Distribution of packet loss outage durations

outages are considerably shorter lived, no doubt because, in $\mathcal{N}_2$, the connections often had many more data packets in flight (§ 9.3), and so had significantly more opportunity to observe short-lived outages.

Figure 15.12 shows the distributions conditioned on the outage exceeding 200 msec, which removes the effect of the $\mathcal{N}_2$ data packets observing more short-lived outages. (The x-axis extends only to 50 sec even though all of the distributions have some larger points. The plotting truncation lets us focus on the main body of the distribution in more detail than we could if we included the entire upper tail.) We see that, for outages of this length or longer, all four distributions agree fairly closely.

It is clear from Figure 15.11 that outage durations span several orders of magnitude. For example, 10% of the $\mathcal{N}_2$ ack outages were 33 msec or shorter, while another 10% were 3.2 sec or longer, a factor of a hundred larger. Furthermore, the upper tails of the distributions are consistent with those of Pareto distributions. Figure 15.13 shows a complementary distribution plot of the duration of $\mathcal{N}_2$ ack outages, for those lasting more than 2 sec (about 16% of all the outages). Both axes are log-scaled, so a straight line on the plot corresponds to a Pareto distribution. We see the long outages fit quite well to a Pareto distribution with shape parameter $\alpha = 1.06$, except for the extreme upper tail, to which we will return in a moment.

A shape parameter $\alpha \leq 2$ means that the distribution has *infinite variance*, indicating immense variability. Pareto distributions for activity and inactivity periods play key roles in some models of self-similar traffic [WTSW95, WP97, WPT97]. We do not attempt further analysis here of the possible role of packet loss outages in contributing to self-similar correlations in aggregate network traffic, but note that it may prove a fruitful area for further research.

However, it is clear in the plot that the extreme upper tail does not fit the same Pareto distribution. This discrepancy could simply be because the uppermost tail is subject to truncation, due to the 600-second lifetime to which our connections were limited (§ 9.3). But the discrepancy could in-

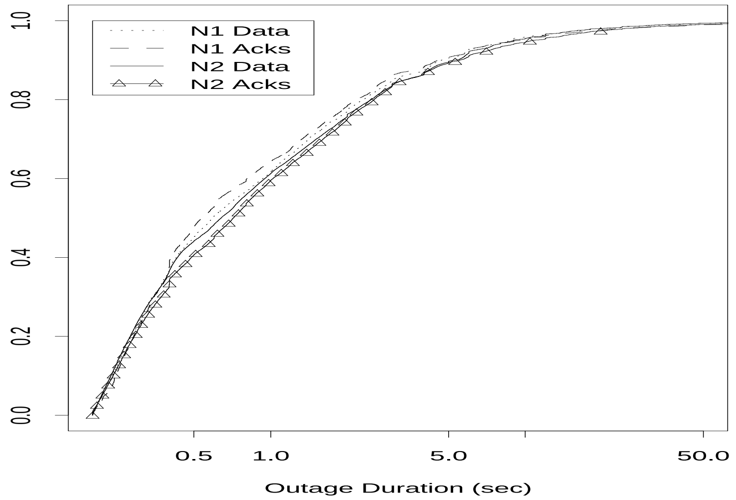

Figure 15.12: Distribution of packet loss outage durations exceeding 200 msec

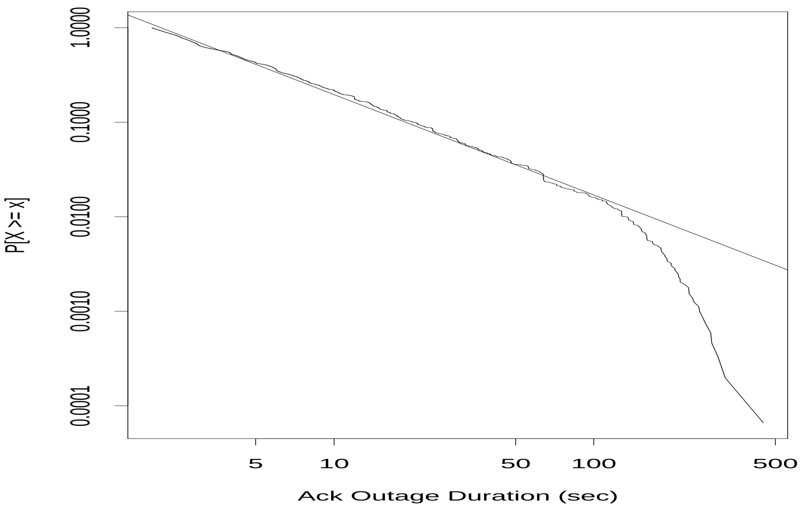

Figure 15.13: Log-log complementary distribution plot of $\mathcal{N}_2$ ack outage durations

stead reflect two different loss mechanisms. We showed in § 6.8 that "temporary outages" observed by `traceroute` measurements appear well-described using *exponential* distributions, which are much less volatile than Pareto distributions. That analysis, however, was confined to time scales of 30 sec or longer, and, for $\mathcal{R}_2$ (corresponding in time to $\mathcal{N}_2$), we found a mixture of exponentials, with the second only fitting to outages exceeding 75 sec in duration. This latter fit corresponds to the extreme upper tail in Figure 15.13. This in turn leads us to speculate that the distribution of outage durations might reflect a Pareto distribution for losses due to heavy congestion, and an exponential distribution for losses due to routing outages. We could test this hypothesis by gathering packet loss measurements made over longer periods of time, which would eliminate the ambiguities presented by the 600-second lifetime truncating the upper tail of our measurements.

We might also consider analyzing the *number* of lost packets in an outage, rather than the duration of the outage. This value, however, is much more subject to fluctuation due to the particulars of how many packets the TCP had in flight prior to the outage, or how many acks it had to generate during the outage in response to incoming data packets. We note that the mean number of packets lost during an outage was around 1.5, slightly lower for acks and higher for data packets. The loss extremes we observed were 68 consecutive data packets and 40 consecutive acks (most of which were dups in response to a large number of incoming packets). These extremes are less interesting than the extreme outage durations, because the former are specific to the structure of the TCP connections—both occurred due to very large numbers of data packets in flight,

We also note that the patterns of loss bursts we observe might be greatly shaped by use of "drop-tail" queueing. With the drop-tail policy, a router queues incoming packets until the available buffer space is exhausted, and then drops any additional arrivals until sufficient space becomes available again. Routers using drop-tail comprise the vast majority of Internet routers, no doubt because it is very simple to implement.

Simulations show that drop-tail leads to large bursts of losses when a flight of closely-spaced packets arrive at a router with no available buffers, and the entire flight is dropped [FJ93]. Related to this problem is a basic *unfairness* in how packets are dropped: a connection may suffer a large number of losses because a *different* connection is occupying all of the router's buffer. In response to these problems, [FJ93] developed the "Random Early Drop" (RED) policy, in which the router drops (or marks) incoming arrivals before all of the buffer has been exhausted. These drops are made with probabilities reflecting the proportion of the router's resources used by the connection, so the policy is much more fair than drop-tail. Because RED *spreads out* losses over time, widespread deployment of RED could significantly alter loss patterns and the corresponding connection dynamics.

A final loss burst pattern we investigated was the presence of *periodic* losses: outages occurring a fixed interval apart. Floyd and Jacobson observed periodic losses and described how they could arise due to global synchronization of the times at which routers exchange updates [FJ94]. They showed how fixed-interval timers such as thirty second update periods act as resonant frequencies, which can synchronize in phase to other events occurring at the same frequency. Periodic losses are thus possibly symptomatic of widespread synchronization in the network, which can have debilitating effects on network performance, especially since large loss periods can in turn synchronize all of the TCP senders that suffer a loss during the period.

Unfortunately, our measurements are ill-suited to detecting periodic loss. Rather than having fixed intervals between our loss "probes" (i.e., the individual packets of a single TCP connection), which would then lend themselves nicely to frequency-domain analysis, we have variable intervals. Furthermore, we used much larger, variable intervals between groups of measurements (connections), precisely to avoid problems with the measurements synchronizing to any periodicities present in the network. Thus, while we can analyze the timing of all of the lost packets in our measurements, the measurements themselves are sparse, and also are cluttered with a great deal of loss that is clearly not periodic.

We attempted to analyze for periodic loss by first identifying a North American subset of our sites with clocks highly synchronized to each other. We identified the day with the most connections between those sites and extracted from the traces a dataset giving the times L_i of each packet loss during those connections. We then constructed plots of L_i versus $L_i \bmod \mu$, and varied μ through the range of $1, 2, \ldots, 120$ sec. We hoped to find a μ for which many of the $L_i \bmod \mu$ clustered about a particular value. However, no compelling modulus emerged. We repeated the analysis for data packets sent to Europe, shown in Table XXII as the most loss-prone Internet path, to test whether perhaps their heavy losses are due in part to a periodic component rather than congestion. Again, we did not find persuasive evidence of frequent periodic losses.

We conclude that periodic losses do not *strongly* dominate TCP packet losses. However, the mismatch between our measurements and those needed to thoroughly examine the question of periodic losses is great enough that we cannot from our evidence conclude that such losses do not regularly occur.

15.4 Loss location

We discussed in Chapter 14 how each network path contains one (or more) "bottleneck" element(s) that limit the maximum rate a connection using the path can achieve. It is natural to assume that this bottleneck element is also the point of congestion along the path, because it has the least amount of one of the network's most important resources, namely bandwidth. Consequently, for a given load in terms of volume of packets to forward along a network path, the bottleneck elements will be the most stressed of those along the path, since they require the most time to service the load. With this assumption, we are again (as in § 15.2) abstracting the intricate, multi-element network path to a presumably equivalent model of a single element that forwards at the bottleneck rate, and at which all significant queueing occurs.

One might think that, with only end-to-end measurements, one lacks sufficient information to verify whether in fact loss occurs at the bottleneck or at some other element. Sometimes, however, we can, as illustrated by Figure 15.14 and Figure 15.15. Both sequence plots reflect data packet arrivals at the receiver, with the packets flowing in steadily at the bottleneck rate. In each plot, one packet has been lost, and the circle indicates where it would have arrived had it not been lost, and had it likewise arrived at the bottleneck rate. In Figure 15.14, its successor arrives in the position where the lost packet would have otherwise arrived. This indicates its successor did *not* queue behind the lost packet, but instead behind the lost packet's predecessor; hence the lost packet must never have made it across the bottleneck link. In Figure 15.15, however, the successor arrives in the same position that it would have, had the lost packet safely arrived too. Thus, the successor *did* queue behind the lost packet at the bottleneck, and we conclude that the lost packet did indeed make it across the bottleneck link, only to be dropped later.

In general, we prefer that packets are dropped *before* the bottleneck, so they do not fruit-

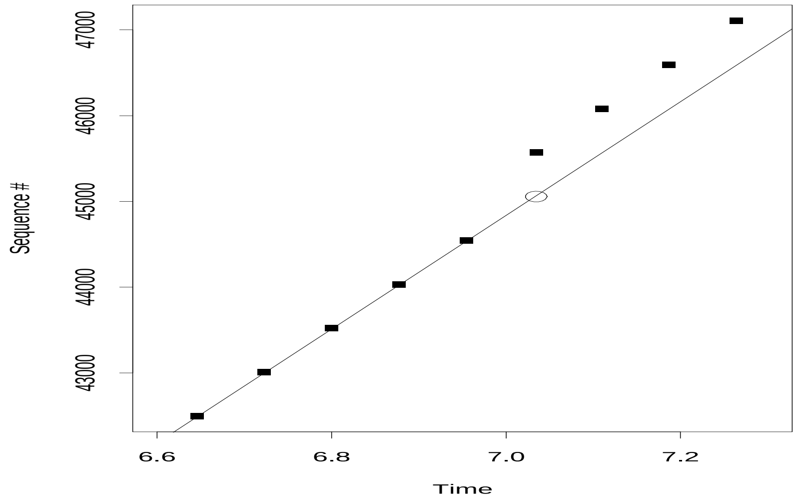

Figure 15.14: Receiver sequence plot showing packet lost at or before bottleneck link

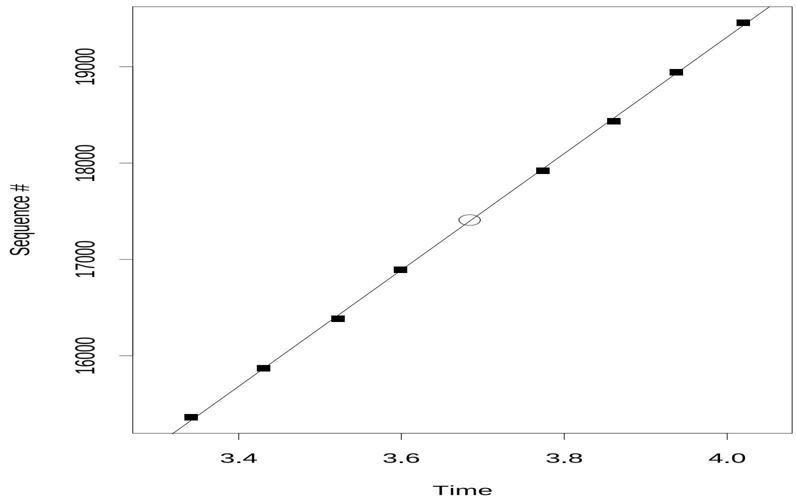

Figure 15.15: Receiver sequence plot showing packet lost after bottleneck link

lessly consume the (usually) scarce bottleneck resource. In this section, we analyze how often this occurs. We first clarify our terminology. We will refer to a packet lost after it has been successfully forwarded by the bottleneck element as occurring "after the bottleneck," while one lost earlier as occurring "before the bottleneck." These latter may have been lost because of a full queue just before the bottleneck element, or may have been lost further upstream. Some network paths may have *multiple* bottlenecks, meaning a number of elements with the same limiting rate. Since our analysis is based on the patterns in Figures 15.14 and 15.15, in the case of multiple bottlenecks we consider only loss after or before the first of the bottlenecks. Loss prior to subsequent bottlenecks will still appear at the receiver as in Figure 15.15, since the data packets will have already been spread out by the first bottleneck.

Our analysis is doomed to be inexact, since effects such as data packet compression (§ 16.3.2) and spurious extra delay often obscure the patterns so clearly evinced in Figures 15.14 and 15.15. But we still aspire to attempt some sort of meaningful analysis, since the basic question of position of loss is an intriguing one, with the potential to reshape our abstractions when analyzing networks.

We proceed as follows. For each lost data packet, we check whether both its predecessor and successor arrived successfully. If not, then we ignore the packet for our analysis, which removes from our possible results the effects of loss bursts. Since we know from § 15.3 that loss bursts are not uncommon, the resulting bias means our results will at best be only qualitative. (We attempted to extend the analysis to include loss bursts, but the ambiguities of whether the next successful packet had to queue behind only *some* of the packets lost in the burst proved too difficult to remove.)

If both predecessor and successor arrived, then we check whether the lost packet was sufficiently "loaded" (§ 15.2) that, upon arriving at the bottleneck, it would find its predecessor waiting in the queue, not yet having begun its service. If not, then we again ignore the packet for our analysis. Doing so assures we only analyze lost packets that would nominally have occupied a full "slot" at the queue, and not a partial slot due to arriving while its predecessor was in the process of transmission across the bottleneck.

If the lost packet was sufficiently loaded, then we check whether its successor was sent soon enough after that, had the lost packet queued at the bottleneck, its successor would have arrived at the bottleneck before the lost packet began *its* bottleneck transmission, and thus the successor would have been delayed a full "slot" in the queue, too. If the successor was sent too late, we again ignore the lost packet for our analysis.

If the successor was sent sufficiently soon after the lost packet, then we next inspect the arrival time of the successor. If it is within $\pm 25\%$ of the time expected had the lost packet never been transmitted (no bottleneck "load" incurred), then we consider the lost packet as having been dropped before the bottleneck. If the successor arrives within $\pm 25\%$ of the time expected had the lost packet indeed loaded the bottleneck, then we consider the loss as occurring after the bottleneck. If the successor's arrival is between these two ranges, then its arrival is "ambiguous," and if its arrival is after (or before) both ranges, then its arrival is "inconsistent," meaning the simple packets-arriving-at-the-bottleneck-rate scenario we envision is inadequate, probably due to downstream queueing.

In both $\mathcal{N}_1$ and $\mathcal{N}_2$, about a third of the losses fit the "inconsistent" category, and almost none were "ambiguous." Of the remaining two-thirds, we find that, in $\mathcal{N}_1$, fully 48% of the losses occurred after the bottleneck. In $\mathcal{N}_2$, the figure falls to 28%. These figures, however, are less than solid in two important ways. First, if a packet is lost before the bottleneck, but its successor queues behind a packet from *another connection* at the bottleneck, then we will still obtain the signature of an after-bottleneck loss. It is difficult to see how to quantify the frequency of this effect given only end-to-end measurement data. Second, our analysis is somewhat skewed by the presence of sites in our study with low-speed Internet connections. For connections involving these sites, the bottleneck will often be immediately at the sender (or before the receiver), so there is little opportunity for loss before (or after) the bottleneck. If we restrict our analysis to only connections with a bottleneck rate exceeding 100 Kbyte/sec, then in $\mathcal{N}_1$ we find 36% of the losses occur after the bottleneck, and 26% in $\mathcal{N}_2$.

From this analysis, we conclude that, for isolated packet losses (not bursts), the assumption that loss occurs at or before the bottleneck link is certainly true more often than not. But if loss position is critical to some analysis, then one must accommodate the possibility of loss occurring after the bottleneck. We also conclude that perhaps 25% of packet loss occurs regretfully late in the network path, meaning that an upstream bottleneck link spent its scarce resources carrying a doomed packet.

15.5 Evolution of packet loss rate

In this section we look at how packet loss rates along an Internet path evolve over time. Our goal is to determine how fruitful it might be to cache packet loss information for Internet paths to better estimate the service we might expect from the paths in the future. For each path in our study, we analyze the evolution of the ack loss rate along the path in several different ways. Clearly, there will be great variation among some of the paths in how the loss rate evolves over time. But we presently limit ourselves to investigating overall patterns of loss rate evolution, aggregated over all of the $\mathcal{N}_2$ connections. We do not analyze the $\mathcal{N}_1$ connections because few of the $\mathcal{N}_1$ paths were measured frequently enough to allow solid analysis.

We first look at how well observing no loss along the path for a 100 Kbyte connection predicts experiencing no loss along the path for another such connection at some point in the future. For each zero-loss connection, c, we compute the pair $\langle \Delta T_c, I_c^z \rangle$, where ΔT_c is the time between that connection and the next successful connection, c', we observed along that path; and I_c^z is an indicator function with a value of 1 if c' also experienced no loss, and 0 if it did.

After constructing these pairs, we sort them on ΔT_c and then compute the probability $P^z(\Delta T)$ that a connection that comes an interval ΔT after a zero-loss connection will also be zero-loss, as follows. Let $I_{(i)}^z$ denote the ith indicator, sorted on ΔT_c. Beginning with $\widehat{P}_0^z = 1$, we run an exponentially-weighted moving average (EWMA) with $\alpha = 0.01$ through the sorted indicators, where the ith value of the average is computed as

$$\widehat{P}_i^z = (1-\alpha)\widehat{P}_{i-1}^z + \alpha I_{(i)}^z.$$

Let $\widehat{P}^z(\Delta T)$ then be $\widehat{P}_i^z$ for the value of i corresponding to the interval ΔT.

Using $\alpha = 0.01$ means that $\widehat{P}_i^z$ is dominated by the preceding 100 values of I^z, though earlier values still contribute to the smoothing. Our goal is to turn the indicator values into meaningful probability estimates, while still allowing for effects that are localized to different time intervals.

Figure 15.16 shows how $\widehat{P}^z(\Delta T)$ evolves with time. The x-axis gives the time between the first zero-loss connection and the subsequent connection, logarithmically scaled, and the y-axis gives the smoothed probability that the subsequent connection is also zero-loss.

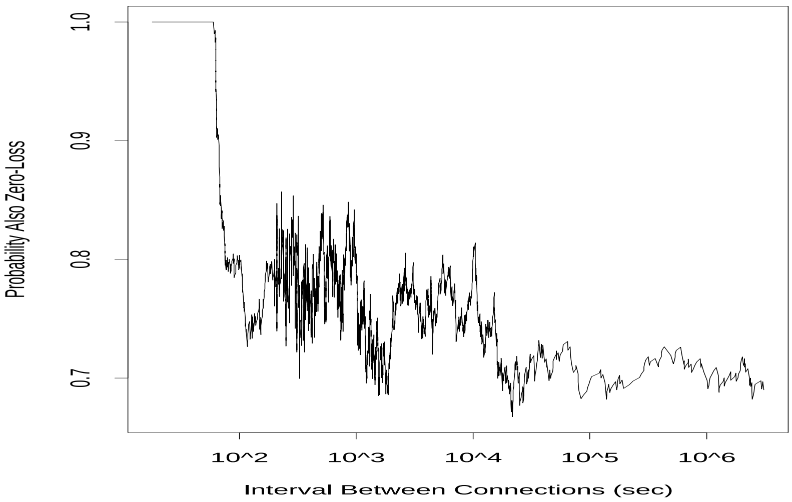

Figure 15.16: Evolution of how well observing a zero-loss connection predicts that a future connection will also be zero-loss

We had very few successive connections in our study separated by less than 60 sec, because the NPDs reuse TCP connection identifiers (to aid in filtering the traffic, per § A.2), and most TCP implementations set a minimum waiting interval on reusing identifiers of 1 minute or more.[6]

Because of the combination of exponential smoothing and very few closely-spaced successive connections, the leftmost portion of the plot exhibits an artifact in terms of a steep dip from probability 1.0 to probability 0.8. Had we instead used an initial probability of $\widehat{P}_0^z = 0.8$, then this spike would disappear. Putting aside the spike, we see that the probability of again observing a zero-loss connection stays at about 0.75 for intervals on the order of a few minutes to a few hours. Above about 6 hours, it approaches what appears to be a "steady state" of 0.70, which continues all the way out to several weeks. Thus, observing a zero-loss connection remains a good predictor of observing future zero-loss connections, even for points in time quite far in the future.

Figure 15.17 shows the same evolution except for the predictive power of observing a non-zero-loss connection rather than a zero-loss connection. The pattern is similar, though the steady state shows signs of declining on time scales of weeks. The "notch" at about 6 hours (21,600 sec) is somewhat puzzling, though it is perhaps simply an artifact, as the region surrounding the notch contains only about 200 points. The notch at four minutes is likewise puzzling: it contains 20% of all of the points, and hence is clearly not spurious, but it is difficult to see what mechanism would lead to less correlation between connections 3-5 minutes apart compared to those further apart. (The comparable notch in Figure 15.16 occurs instead at two minutes, and contains only 3% of the points, so it is perhaps spurious.)

The final aspect of packet loss evolution we look at is how loss rates change over time. For each connection, we compute $\langle T_c, \lambda_c \rangle$, where T_c is the time when the connection began and λ_c

[6]The TCP specification sets this time at 4 minutes, though it provides exceptions for which it can be bypassed [Br89].

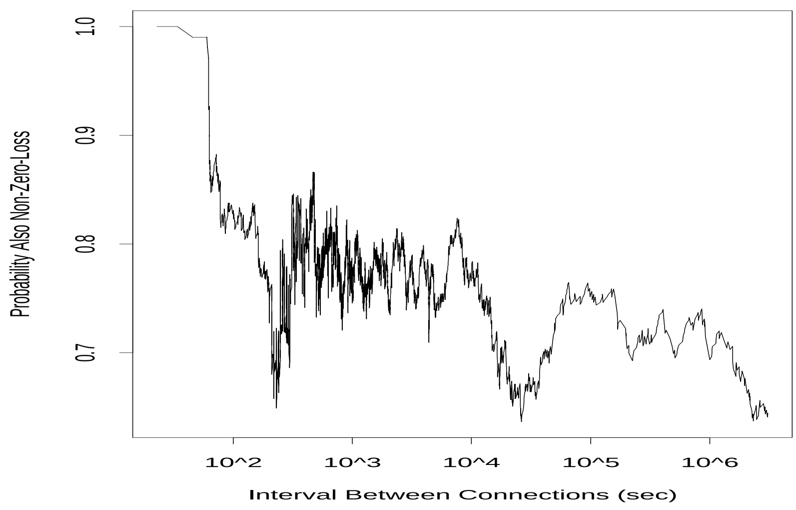

Figure 15.17: Evolution of how well observing a non-zero-loss connection predicts that a future connection will also be non-zero-loss

is the ack loss rate. We then compute for consecutive connections c_1 and c_2 along the same path the pair $\langle \Delta T_{1,2}, \Lambda_{1,2} \rangle$, where:

$$\begin{aligned} \Delta T_{1,2} &= T_{c_2} - T_{c_1}, \\ \Lambda_{1,2} &= |\lambda_{c_2} - \lambda_{c_1}|. \end{aligned}$$

Thus, $\Lambda_{1,2}$ gives the magnitude of the difference in loss rates between the two connections.

Figure 15.18 shows how the EWMA of $\Lambda_{1,2}$ evolves as $\Delta T_{1,2}$ increases, where the smoothing is done with $\alpha = 0.01$ and with an initial value of $\Lambda_{0,0} = 0$. We see an almost immediate jump to a mean difference of $\pm 2\%$ in loss rate, followed by a steady climb up to a difference of $\pm 4\%$ at about 10 hours, followed by a jump to the $\pm 6 - 8\%$ level for larger time intervals, where the variation for very large time scales (weeks) at the righthand edge of the plot may be spurious, due to an exceedingly small number of samples.

From Figures 15.16, 15.17, and 15.18, we conclude that observing no loss along a path is a good predictor that we will continue to not observe loss along the path, even far into the future; that the same holds almost as strongly for observing loss predicting we will observe future loss; but that the farther into the future we wish to project, the more difficult it is to accurately assess the *magnitude* of the loss rate based on the magnitude of the currently observed loss rate. These findings support the notion developed earlier in this chapter that network paths have two general states, a tendency towards loss-free connections (“quiescent”), and a tendency towards lossy connections (“busy”), and provide evidence that both states are long-lived, on time scales of hours, presumably because they are functions of whether the path has adequate capacity for the aggregate traffic delivered to it, and aggregate traffic rates generally change on time scales of hours [PF95]. We also find that, while we may predict future loss rates fairly accurately for time scales of minutes to hours, as time scales grow beyond, our predictive power diminishes.

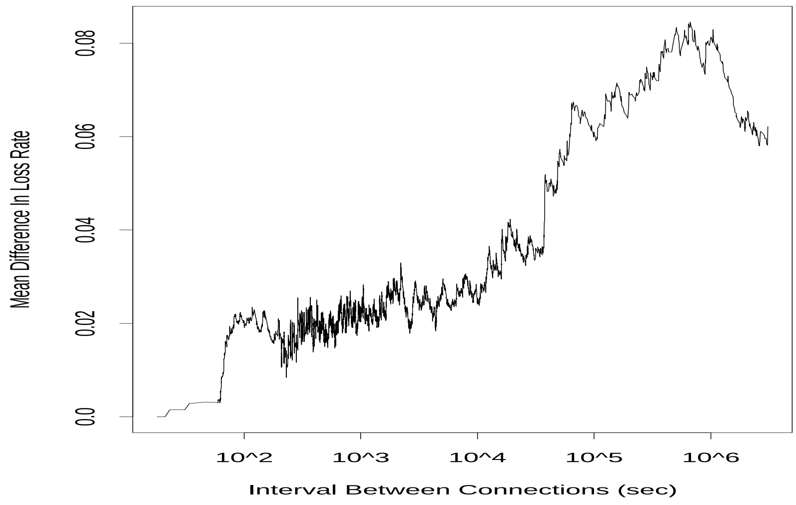

Figure 15.18: Evolution of the mean difference in loss-rate between successive connections along the same path

15.6 Efficacy of TCP retransmission

The final aspect of packet loss we investigate is how efficiently TCP deals with it. Ideally, TCP retransmits any lost data until it is successfully received, but never retransmits unnecessarily, as that would waste network resources. However, the transmitting TCP lacks perfect information, and consequently will sometimes indeed retransmit unnecessarily. For example, TCP acknowledgements are *not* transmitted reliably; so, if a flight of data packets all arrive successfully at the receiver, but all of the corresponding acknowledgements are lost, then the TCP has no choice but to retransmit when the retransmission timer expires.

We analyzed the efficacy of retransmission by the different TCPs in our study as follows. For each connection, we examine each retransmitted packet P_r to see if the data contained in P_r had already been successfully sent.[7] Note that the earlier, successful transmission may not have arrived yet at the receiver at the time of the retransmission; we consider it successful, however, if an earlier transmission of the data *ever* arrives at the receiver.

If P_r contained data that had not previously been successfully transmitted to the receiver, then we term P_r "necessary," otherwise we term it "redundant." *In both $\mathcal{N}_1$ and $\mathcal{N}_2$, about 40% of the retransmissions were redundant!* As an aggregate statistic, this is not a happy number. It means that two times out of five, the TCP should (1) not have retransmitted, and (2) not have cut its congestion window, if the retransmission led it to do so. However, we need to investigate the 40% figure better, since there are a number of different reasons why a TCP might send redundant retransmissions (RRs):

[7]The exact test is whether *all* of the data in P_r had been successfully sent. This fine point can be important if different portions of P_r's data were earlier sent in different packets.

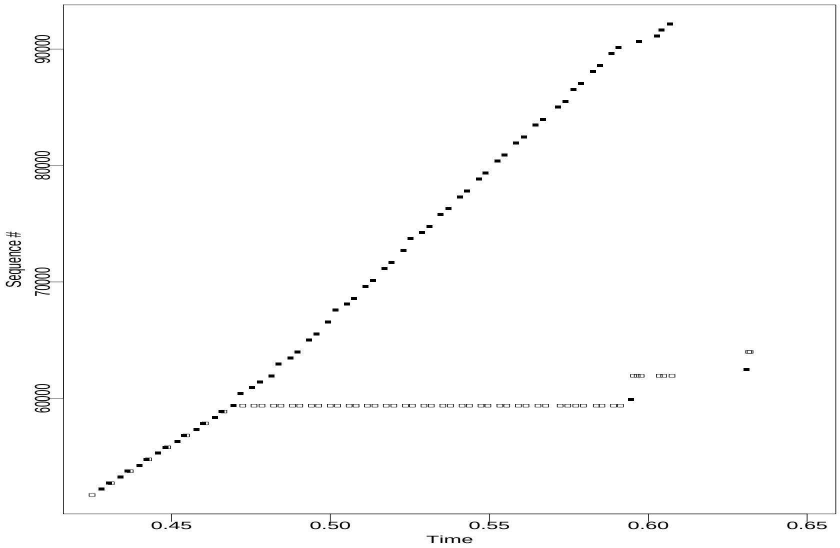

Figure 15.19: Receiver sequence plot showing large number of sequence holes

unavoidable We mentioned earlier that, if the network drops all of the acks for a flight of data packets, then the TCP sender has no choice but to retransmit, since no further feedback will be forthcoming from the receiver.

pathological The packet was a timeout retransmission, but the interval between the data's earlier transmission and this packet's was less than the minimum round-trip time ever seen. Hence, the retransmission timeout used by the TCP was absolutely broken—the receiver did not even have a chance to acknowledge the data—and, furthermore, a simple test by the TCP to make sure that at least the minimum RTT had elapsed would have prevented the redundant retransmission.

coarse feedback Since TCP acknowledgements simply give the highest data sequence number received in-order, when a TCP retransmits with a window larger than one packet (such as during slow-start after a timeout), it may transmit unnecessary packets because the receiver lacks a fine enough feedback mechanism to tell it which above-sequence packets have already arrived. Figures 15.19 and 15.20 illustrate the problem. In the first sequence plot (measured at the data receiver), we see that the sender has a large amount of data in flight, which until about $T = 0.47$ has steadily streamed in. At that point, however, the packet with sequence number 59,905 is lost. Many more packets continue streaming in, but they contain numerous holes where some were lost. The new arrivals generate a torrent of duplicate acks in response. Since, however, the acks only provide coarse feedback to the sender, all the sender really knows is that sequence 59,905 was lost, and many more packets safely arrived—but it does not know which.

The sender retransmits the first missing packet via fast retransmission (§ 9.2.7), and this packet arrives at the receiver just before $T = 0.6$. The receiver duly acknowledges up to the next hole, and even generates some duplicate acks for new data arriving at sequence 90,625

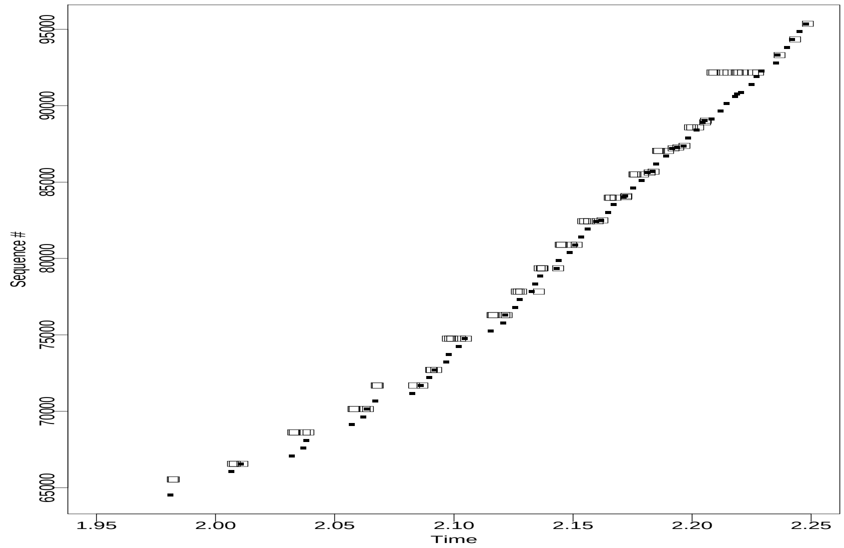

Figure 15.20: Redundant retransmissions subsequent to previous figure

and above (sent due to fast recovery). These in turn lead to a fast retransmission for the next hole, arriving at $T = 0.63$. At this point, however, the sender does not see any more incoming acks allowing it to send more data via fast recovery (and it has halved its congestion window twice, once per fast retransmission event, so it will take a while for more dup acks to inflate the window far enough to enable fast recovery). Consequently, self-clocking ceases and the sender stalls until a retransmission timeout occurs.

Until now, the retransmissions have all been necessary. The retransmissions after the timeout, however, are a disaster, as shown in Figure 15.20. The first packet retransmitted after the timeout was also necessary. Unfortunately, the acks generated by it (shown as large squares in the plot) rapidly open the sender's congestion window due to slow start, and it sends larger and larger flights of packets. Nearly all of these retransmitted packets are unnecessary—all that is really needed is to fill the sporadic holes shown in Figure 15.19. Every duplicate ack in Figure 15.20 corresponds to an unnecessary retransmission, yet because the sender lacks fine-grain information regarding which above-sequence packets the receiver already has, it continues retransmitting to fill the known holes (as indicated by the latest ack it has received), as well as pouring additional, unnecessary packets into the network—23, all told.

The TCP research community has long known about this problem and is in the midst of standardizing a TCP extension to remedy it. With the extension, a "selective acknowledgement" (SACK) option, acks can carry additional information concerning above-sequence packets that have arrived at the receiver (§ 13.1.3). The sender then uses this information to select which packets require retransmission.

We consider an RR as reflecting TCP's "coarse feedback" problem if it occurred *after* the arrival of an ack that itself was sent after the original copy of the data arrived at the receiver. Presumably, had we used SACK, this ack could have conveyed to the sender that the data had

Type of RR	$\mathcal{N}_1$ total	$\mathcal{N}_2$ total	$\mathcal{N}_1$ Solaris	$\mathcal{N}_2$ Solaris	$\mathcal{N}_1$ Other	$\mathcal{N}_2$ Other
% all packets	2%	3%	6%	6%	1%	2%
% retransmissions	43%	38%	66%	59%	26%	28%
Unavoidable	25%	25%	14%	33%	44%	17%
Pathological	2%	7%	3%	11%	0%	2%
Coarse feedback	18%	41%	1%	1%	51%	80%
Bad RTO	55%	28%	81%	55%	4%	1%

Table XXIV: Proportion of redundant retransmissions (RRs) due to different causes

already arrived, and the sender would have avoided the RR.

bad RTO If the RR was prompted by a timeout, and if an acknowledgment for the previously sent data arrives after the timeout retransmission, then the TCP selected too low a value for its retransmission timeout (RTO). The RR could have been avoided simply by waiting longer.

Table XXIV summarizes the prevalence of the different types of RRs in $\mathcal{N}_1$ and $\mathcal{N}_2$. The second and third columns give the overall percentage of the $\mathcal{N}_1$ and $\mathcal{N}_2$ RRs due to each type. The fourth and fifth columns give the same figures if we restrict the analysis to just Solaris TCP senders, since in § 11.5.10 we discussed how it is prone to underestimating RTO and consequently retransmitting too early, so we would expect it to exhibit a higher frequency of "pathological" and "bad RTO" types of RRs than the other TCPs in our study. The final two columns summarize the frequency of each type of retransmission for the non-Solaris TCPs.[8]

We see that a fair proportion of the RRs were unavoidable. (Some of these might, however, have been avoidable had the receiving TCP generated more acks.) We note that for $\mathcal{N}_2$, which, with its bigger windows (§ 9.3), had more opportunity to successfully transmit an ack for part of the window, only about 1/6 of the RRs for non-Solaris TCPs were unavoidable. Clearly it is worth our efforts to first eliminate the avoidable 5/6's.

Pathological RRs could be eliminated with a simple test: if the packet being retransmitted was previously transmitted (or retransmitted) less than one RTT in the past, then simply do not retransmit it. Aside from Solaris, most pathological RRs occur within retransmission epochs, during which earlier RRs lead to enough duplicate acks that the TCP resends data it sent shortly before due to the window advancing. For Solaris, many occurred due to the problems the Solaris TCP timer has with adapting to the true round-trip time, cf. § 11.5.10 and § 11.5.1.

"Coarse feedback" RRs would presumably all be fixed using SACK. The increase in non-Solaris coarse feedback RRs in $\mathcal{N}_2$ is no doubt due to the use of bigger windows in $\mathcal{N}_2$, and hence more opportunity for acks (and, thus, finer feedback) to potentially inform the sending TCP of what

[8] In § 11.5.8 we identified the Linux 1.0 TCP as suffering from many RRs due to its practice of retransmitting all the unacknowledged packets rather than just the first. However, in § 10.5 we discussed how many of the Linux traces could not be unambiguously paired in terms of packet departures and arrivals, precisely because of this retransmission problem. In this section, we confine our retransmission analysis to those traces that we could unambiguously pair, so we can distinguish between the different types of RRs (in particular, "coarse feedback," which depends on whether the original data arrived before a subsequently transmitted and received ack). Consequently, we analyzed very few Linux 1.0 traces and thus their presence does not significantly affect the statistics in Table XXIV.

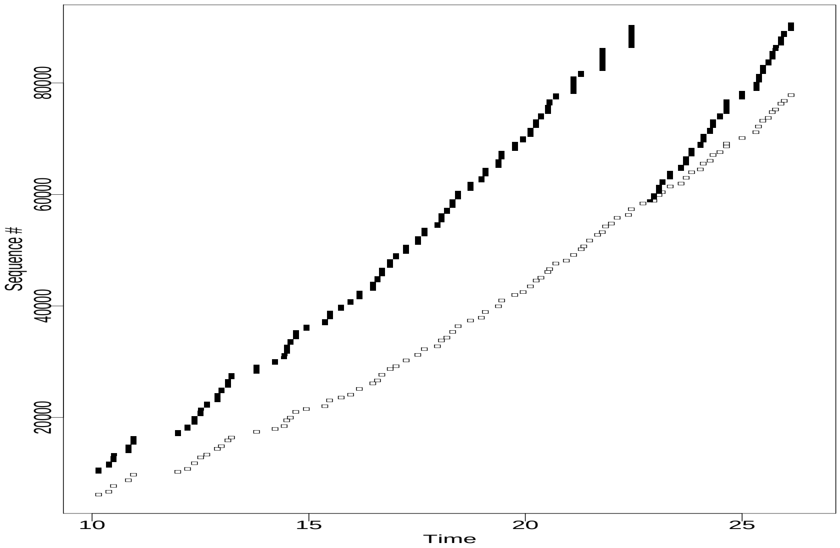

Figure 15.21: Sender sequence plot showing failure of RTO adaption

packets the receiver already has. It is encouraging to see that, aside from Solaris TCPs, deployment of SACK remedies almost all of the avoidable RRs. It makes almost no difference for Solaris TCP, since many of its RRs occur before any ack for the previous transmission of data has arrived from the receiver, due to the Solaris timer adaption problems.

"Bad RTO" RRs indicate that the TCP's computation of the retransmission timeout was erroneous. These are the bane of Solaris TCP, as noted above. More than half of its RRs were due to miscalculating the timeout. Fixing the calculation eliminates 4-5% of *all* of the data traffic generated by the TCP.[9]

The TCP standard requires use of Jacobson's exponentially-weighted moving average (EWMA) round-trip time (RTT) estimate and associated variance estimate ([Br89, 4.2.2.15] and [Ja88]), along with Karn's algorithm for eliminating ambiguous RTT estimates [KP87]. If we assume that the non-Solaris TCPs do in fact implement this algorithm, then from Table XXIV we see that it performs quite well.

Figure 15.21 shows an instance where it failed, or at least where HP/UX 9.05's implementation of it failed. Here the receiving TCP is offering a very large window, to which the sending TCP is rapidly opening its congestion window in the face of no packet loss. The bottleneck link, however, can only support about 7.3 Kbyte/sec, and so the window represents a large mismatch with the correct window size needed to avoid overloading the bottleneck. Consequently, the RTT rises rapidly as packets queue behind their predecessors. During the last five round trips, starting at time $T = 10$, the RTT increases by about 1 second during each trip. The RTO estimation algorithm fails to track this rapid increase, and at time $T = 23$ a retransmission timeout occurs, even though the corresponding ack is just about to arrive. Subsequent acks for the first transmissions of the data then rapidly feed the slow-start sequence begun by the timeout retransmission, and the sending TCP

[9] We note that this problem has already been fixed in Solaris 2.5.1.

promptly resends 63 packets, all redundant. However, we found pathological behavior like that shown in the figure quite rare.

While the standard RTO estimation almost never leads to an unnecessary timeout retransmission, a separate question, unanswered by these statistics, is whether it could be safely modified to be less conservative. At present the timeout often occurs after much more than an RTT elapses. A more aggressive RTO algorithm could potentially lead to higher connection throughput, because timeout lulls would be less costly than they currently are. Yet, if the more aggressive algorithm leads to excessive retransmission during times of RTT fluctuation, then it could contribute to congestion collapse, a major disaster.

Answering the question of how the RTO estimation might be reengineered is a complex problem. The current timer uses coarse-grained (as much as 500 msec granularity) measurements with some subtle adjustments to compensate for the granularity, as well as timing only one packet per flight. A revised timer might take advantage of both higher-resolution clocks and the opportunity to time multiple packets per flight. The first affects the adjustment factors used by the current algorithm, and the second changes the constant used in the EWMA estimator. Because the issues are complex, we leave this interesting question for future work.

In summary: assuring standard-conformant RTO calculations and deploying the SACK option together eliminate virtually all of the avoidable redundant retransmissions. The few remaining RRs are rare enough to not present, overall, any serious performance problems.

The last aspects of TCP retransmission we investigate are the patterns of packet loss during fast recovery sequences. The TCP fast recovery mechanism, described in § 9.2.7, works best when only a single packet out of a flight is lost. When multiple packets in one flight are lost, the fast recovery mechanism generally will not suffice to retransmit all of the missing packets, and the TCP transfer will subsequently stall until a retransmission timeout, seriously diminishing throughput [FF96, Ho96]. It was this problem that motivated the development of the SACK option, which allows a TCP to efficiently recover from multiple losses.

A separate fast recovery problem occurs when the retransmitted packet is also lost.[10] When this happens, the TCP will again stall until a retransmission timeout expires. In some circumstances, and depending on the algorithm used by a TCP to act upon information it acquires by using the SACK option, a TCP using SACK can avoid this timeout by determining that the retransmitted packet was itself lost, and retransmitting it again.

While these problems have been recognized for quite a while, no hard data has been available in order to gauge the degree to which they actually present difficulties for Internet connections. We analyzed the $\mathcal{N}_1$ and $\mathcal{N}_2$ measurements to provide such data, as follows. For each packet retransmitted using the fast recovery mechanism, we tallied whether the retransmitted packet was lost or successfully arrived at the receiver, and also counted the number of outstanding (unacknowledged) packets at the time of the retransmission that were lost.

In $\mathcal{N}_1$, out of 1,178 packets retransmitted using fast recovery, only 3.9% were themselves lost. In $\mathcal{N}_2$, 15,444 packets were retransmitted using fast recovery (a significantly higher proportion of all of the retransmissions than in $\mathcal{N}_1$, due to the use of bigger windows in $\mathcal{N}_2$, per § 9.3). Of these, only 4.5% were also lost. (These proportions are quite close to the unconditional loss rates we examined in § 15.1, and much lower than the conditional loss rates examined in § 15.3, indicating

[10] This problem also occurs for TCPs that implement "fast retransmit" (§ 9.2.7) but not fast recovery. However, for simplicity, we will only use the term "fast recovery" in our discussion.

that congestion often drains on time scales of RTTs.) Thus, we conclude that the second concern discussed above is, in practice, not an especially serious problem.

However, in both $\mathcal{N}_1$ and $\mathcal{N}_2$, one third of the time more than one packet was lost in the flight prior to a fast recovery, and about 15% of the time, more than two packets were lost. These proportions are high enough to give solid support for refining the fast recovery mechanism (such as by adding SACK, or the modifications discussed by Hoe [Ho96]) in order to better cope with multiple packet losses within a single flight.