where δ is a fraction between 0 and 1. That is, we calculate both the mean RTT and the variation in that mean.

TCP then computes the timeout value as a function of both EstimatedRTT and Deviation as follows:

$$\text{TimeOut} = \mu \times \text{EstimatedRTT} + \phi \times \text{Deviation}$$

where based on experience, μ is typically set to 1 and ϕ is set to 4. Thus, when the variance is small, TimeOut is close to EstimatedRTT; a large variance causes the Deviation term to dominate the calculation.

Implementation

There are two items of note regarding the implementation of timeouts in TCP. The first is that it is possible to implement the calculation for EstimatedRTT and Deviation without using floating-point arithmetic. Instead, the whole calculation is scaled by 2^n, with δ selected to be $1/2^n$. This allows us to do integer arithmetic, implementing multiplication and division using shifts, thereby achieving higher performance. The resulting calculation is given by the following code fragment, where $n = 3$ (i.e., $\delta = 1/8$). Note that EstimatedRTT and Deviation are stored in their scaled-up forms, while the value of SampleRTT at the start of the code and of TimeOut at the end are real, unscaled values. If you find the code hard to follow, you might want to try plugging some real numbers into it and verifying that it gives the same results as the equations above.

```
{
    SampleRTT -= (EstimatedRTT >> 3);
    EstimatedRTT += SampleRTT;
    if (SampleRTT < 0)
        SampleRTT = -SampleRTT;
    SampleRTT -= (Deviation >> 3);
    Deviation += SampleRTT;
    TimeOut = (EstimatedRTT >> 3) + (Deviation >> 1);
}
```

The second point of note is that the Jacobson/Karels algorithm is only as good as the clock used to read the current time. On typical Unix implementations at the time, the clock granularity was as large as 500 ms, which is significantly larger than the average cross-country RTT of somewhere between 100 and 200 ms. To make matters worse, the Unix implementation of TCP only checked to see if a timeout should happen every time this 500-ms clock ticked, and would only take a sample of the round-trip time once per RTT. The combination of these two factors could mean that a timeout would happen 1 second after the segment was transmitted. Once again, the extensions to TCP include a mechanism that makes this RTT calculation a bit more precise.