Agressive Transmissions over Redundant Paths for Time Critical Messages †§

Hector Garcia-Molina
Department of Computer Science
Stanford University
Stanford CA 94305

Ben Kao
Department of Computer Science
Princeton University
Princeton, NJ 08544

Daniel Barbará
MITL
Princeton, NJ 08544

ABSTRACT

Fault-tolerant computer systems have redundant paths connecting their components. Given these paths, it is possible to use aggressive techniques to reduce the average value and variability of the response time for critical messages. One technique is to send a copy of a packet over an alternate path before it is known if the first copy failed or was delayed. A second technique is to split a single stream of packets over multiple paths. We analyze both approaches and show that they can provide significant improvements over conventional, conservative mechanisms.

---

† This paper is a revised and extended version of [7]. The main differences from [7] are the inclusion of Section 4 and the Appendix (containing proofs).

§ This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and by the Office of Naval Research under Contracts Nos. N00014-85-C-0456 and N00014-85-K-0465, and by the National Science Foundation under Cooperative Agreement No. DCR-8420948. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

1. Introduction

In many applications, computer systems have redundant communication paths to improve reliability. At the local level there may be multiple busses or local area networks connecting processors. For example, a Tandem cluster has a dual high speed local broadcast channel[9]. At the network level, in most cases there are at least two independent paths between every two points, so that a single failure cannot partition the network. At the application level, it is also common to have multiple links between critical components. For example, a bank with two main processing centers may have two (or more) separate connections, each provided by a different vendor. As the January 15, 1990 failure of the AT&T network illustrated[10], complete networks may fail, so it is desirable to have a fully independent alternative. In this paper we deal exclusively with reliable systems, and hence assume that redundant paths (of one type or another) exists.

Traditionally, redundant paths are utilized by the communication software in a *conservative* fashion: Given a particular message or packet to be sent to a destination, the source computer or switch selects, from the available paths, the one expected to perform best. (Statistics of previous response times may guide the decision.) The packet or message is sent along the selected path. If problems arise (e.g., no acknowledgement arrives within a timeout period), an alternate path is selected and the packet or message is re-transmitted. An even more conservative approach is to keep one of the paths unused as a hot standby, and to switch all traffic to it when the primary path fails. This approach is more likely to be used for application level redundant paths.

In this paper we explore more *aggressive* approaches. One simple idea is to split a stream of packets over the available paths to improve response time. The goal is to use the extra bandwidth available in no failure periods. A second, less obvious idea is to send redundant copies of the packet or message over independent paths at once, without waiting for a failure or a timeout to occur. This also reduces the expected message delivery time for two reasons. First, since multiple copies are sent, the probability that one makes it in the first attempt is much higher, thus costly timeout and re-transmissions may be avoided. Second, if the transmission time over one network is variable (due to channel collisions, congestion, switch delays, etc.), then the overall delay will be reduced, as the destination only has to wait for the *fastest* copy to arrive. Note that not only are expected response times reduced, but the *variability* in response time is also reduced.

Incidentally, redundancy in data communications is not entirely new. Error correcting codes are sometimes added to packets to enable reconstruction of lost bits. This type of redundancy improves response times by making it unnecessary to re-transmit when a few bits are lost. We use the same principle, except that we send the redundant data over existing *independent* paths to avoid delays and failures associated with a single path. Ramanathan and Shin [19] have also studied the use of multiple copies in real-time systems. While their studies focuses on minimizing the expected cost incurred as a result of messages missing their deadlines, our work addresses the response time speedup obtainable through the agressive approaches. Also, we study the case of 100% redundancy (2 full copies of packet), at least in this initial paper. We will discuss shortly how redundancy can be reduced while retaining some of the speedup benefit.

In order to convince the reader that aggressive transmissions are a good idea, we must address three critical questions: (1) Is the cost reasonable? That is, by doubling transmissions or splitting streams, will we not create more load and cause delays that will offset gains? (2) Why are improved response time and reduced variability important? (3) What exactly are the gains we can expect in response time? This last question is the main focus of this paper. As we will see, there are cases where gains can be very significant, but there are other cases, depending on the communication protocol used and the message size, where they are not. Thus, a careful and realistic evaluation of the potential gains is important. Before going into this topic, we address the first two questions.

To argue that the cost of redundant transmissions is reasonable, we first note that we are *not* proposing duplication of *all* transmissions. We are focusing on a small fraction of time critical transmissions, so that the overall load increase may not affect the response time of transmissions over single paths significantly. Furthermore, in most dependable applications, there is already an excess of communication capacity. The system should continue to operate with acceptable performance even when one or more of the links have failed. This means that during normal operation there must be extra bandwidth. We wish to utilize this spare capacity during normal operation to provide more than just the bare minimum acceptable service. Furthermore, communications cost (dollars/bit) is dropping rapidly with new technology (e.g., optical links). This means that redundant transmissions may be cost effective, just like mirrored disks (redundant storage) and processor pairs (redundant computing) are. Similarly, the extra CPU cost of sending and receiving duplicate messages can be justified by dropping processor costs and the transmission time improvements we will show in this paper. Finally, we note that "partial redundancy" is feasible. For example, if three paths are available, we can send one packet via one path, a second packet via the second, and a parity packet via the third (see Figure 1). Any two of the arriving packets can be used to reconstruct the two data packets. This reduces the overhead, while still achieving some of the response time benefits. We do not analyze this partial redundancy case in this paper, but we do mention it as an area of future research. Arguing for split streams is easier, since all that is involved is higher processing and buffering requirements. Again, given current hardware trends, the extra resources may be well invested.

A faster and more predictable response time is important for time-constrained applications[12] where messages must be delivered within a certain time limit, or they will be useless. Examples of such applications include packetized voice[2], distributed sensor networks, and other real-time control applications[21].

Conventional applications can also benefit from reduced response times. Message transmission time represents a major portion of the total execution time of many distributed algorithms, and a reduction in response time can therefore provide a significant speed up. For example, in mutual exclusion[14, 15], commit[8], and election[6] algorithms running time is dominated by message delays, since there is relatively little processing.

In some cases, the response time gains can be magnified substantially due to reduced resource contention. For example, during a distributed commit protocol in database applications, participants typically hold locks on modified database objects. With reduced network response times and consequently faster commit protocols, resources are held for a shorter period of time, alleviating contention. As a result, total system throughput is improved.

A smaller variability in response time is another source of potential gains. For example, in order to achieve accurate clock synchronization, a transmission time with small variability is required. Many clock synchronization protocols[20] have upper bounds on synchronization accuracy inversely proportional to the variability of message transmission delay. Therefore, a network with a more predictable latency gives us tighter clock synchronization. This has important implications in areas like distributed agreement and consensus protocols[18]. In particular, many consensus protocols assume a synchronous model — processors with closely synchronized clocks.† Moreover, these algorithms are usually designed to be carried out in rounds (made possible by their synchronized clocks), with the duration of each round being equal to the maximum send-receive-forward time — the time taken by each processor to broadcast a message, to receive all incoming messages, and to perform some computation. It has been shown that[4] in a synchronous fail-stop model, the worst case time complexity is $k + 1$ rounds where k is the upper bound on the number of failures tolerated. A shorter response time together with less variability enable these consensus protocols to have shorter rounds, immediately giving us faster algorithms. Finally, with a more accurate global snapshot obtainable using closely synchronized clocks, we can reduce the likelihood of the occurrences of anomalies in a distributed environment such as those suggested by Lamport[13] and phantom deadlocks[17].

†Without clock synchronization (i.e., for asynchronous models), it has been shown that no protocol exists for reaching agreement in the fail-stop model that can tolerate a single failure[5].

The rest of the paper is organized as follows. In the next section, we present our model for the end-to-end round trip latency of a network. In sections 3, 4 and 5 we analyze the response time speed up that aggressive strategies provide for single packet message, medium sized message, and long message transmissions respectively. Finally, section 6 offers some conclusions.

2. The Model

Our goal is to evaluate the response time gains obtained by sending packets over multiple networks. We assume we have a source node transmitting data to a destination node. We study and compare two scenarios: (a) the source and destination are connected by two independent networks, i.e., a 2-net, and (b) they are connected by a single conventional network, a 1-net. In this paper we will analyze three cases: a short one-packet message (Section 3), a medium sized message that consists of several packets (Section 4), and a long message transmitted using a sliding window protocol (Section 5).

Before analyzing the effect of flow control protocols and of duplicate transmissions, we need to understand the basic performance of the underlying communication medium. That is, we need to model the distribution of the *round trip delay* between the source and the destination nodes for a single packet, as well as the probability that the packet gets lost. This base model does *not* take into account re-transmissions performed by the source when a packet is not acknowledged; such effects are incorporated later. We would like our model to be general enough so it can be applied to any type of network, either wide-area, point-to-point or local-area multiple access (see Figure 2).

If we think of the source-destination round trip latency as a random variable, what probability density function (pdf) should it have? We believe it should have the shape shown in Figure 3 (experimental data [3] supports this). There is a minimum transmission time dictated by the speed of light and the processing time at intermediate switches (if any). The probability that the transmission time is less that this minimum is zero. Beyond this limit, there is a range of times that are the most likely to occur, but larger delays may also occur. These long transmission times arise due to congested or faulty switches in a wide-area network or due to collisions in a local area network. Some messages may never arrive, but we model this in our next step.

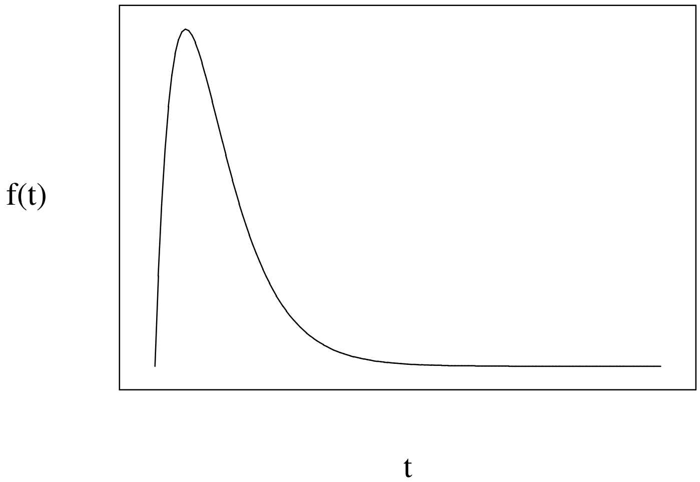


Figure 3 : End-to-end round trip delay

For our analysis we desire a function that has the shape of Figure 3 and is also relatively easy to work with. Thus, we chose a shifted two-stage hypoexponential random variable with parameters $1/\lambda_1$ and $1/\lambda_1+1/\lambda_2$ to represent the underlying round trip latency of a single network. Its probability density function (pdf) is given by :

$$f(t) = \begin{cases} \dfrac{\lambda_1 + \lambda_2}{\lambda_1^2}\,(1 - e^{-(t-\Delta)/\lambda_2})e^{-(t-\Delta)/\lambda_1} & t > \Delta, \\ 0 & t \leq \Delta, \end{cases}$$

where Δ is the minimum delay. By varying the values of λ_1 and λ_2, we can model networks with different transmission delay characteristics: The smaller the value of λ_1, the faster $f(t)$ rises at $t = \Delta$. The smaller the value of λ_2, the faster $f(t)$ falls as t grows. Choosing $\Delta = 1$ time unit, Figure 4 shows curves of $f(t)$ for various values of λ_1 and λ_2. From this figure, one can see that our model can cover a wide range of networks, either slow or fast, with low or high variance.

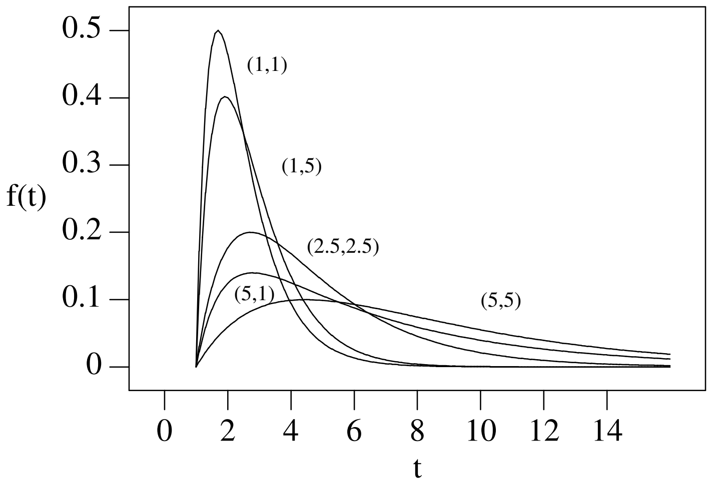


Figure 4 : f(t) for various (λ_1, λ_2) pairs

Another argument in favor of the hypoexponential distribution is that it arises from a series of exponential distributions. That is, if the network is a point-to-point and if we approximate the IMP-IMP transmission delay by an exponential (or hypoexponential) distribution with some minimum delay, then it can be shown that the overall end-to-end round trip latency of the network is hypoexponentially distributed with some minimum delay[23].

To model lost packets we truncate our distribution $f(t)$ at a constant T_2. In other words, we assume that packets that take more than T_2 to be acknowledged will *never* be acknowledged. The probability that a packet makes it, p, is given by

$$p = \int_0^{T_2} f(t)dt.$$

With probability $1 - p$ a packet will not be delivered. Figure 5 shows the relationship between p and T_2. Note that assuming a cutoff time T_2 is reasonable, as most networks that can store packets internally do discard them if they loiter for a long time (this is also required by some transport layer protocols for proper connection management)[22].

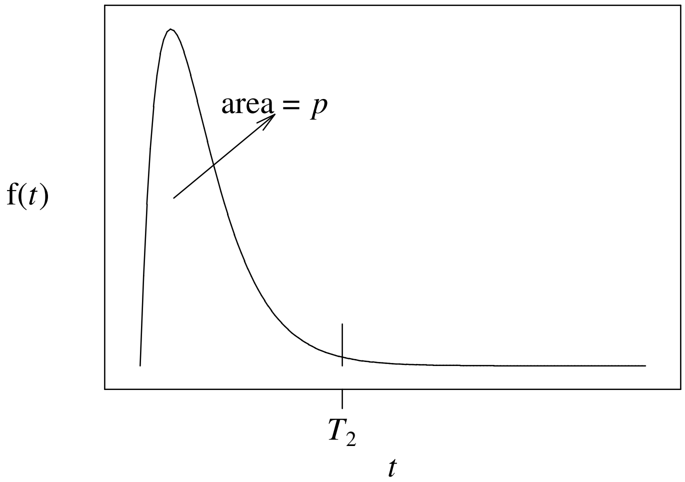


Figure 5 : end-to-end round trip delay

Given our assumption, it makes no sense at all for the source node to set a timeout interval larger than T_2. If the packet does not arrive by T_2, it never will. Conversely, if timeout is set smaller than T_2, the source node risks the possibility of sending a redundant packet and this complicates our analysis. Therefore, we assume that the source node sets its timeout interval to be equal to T_2. (In reality, messages can arrive after the sender times out. Our computed transmission times are hence slightly overestimated, but the difference is small given that very few messages show up after a time out. Furthermore, the response time reduction is greater for a 2-net, so the speed up values we compute later are pessimistic, i.e., 2-nets perform *better* than what we indicate.)

If we let X be the conditional random variable denoting the round trip latency of a single network given that *the packet is not lost*, then X has its pdf, $h(t)$, given by

$$h(t) = \begin{cases} \frac{1}{p} f(t) & t \le T_2, \\ 0 & t > T_2. \end{cases}$$

Note that

$$Pr[X > t] = \int_t^{\infty} h(t)dt \qquad \text{and,}$$

$$E[X] = \int_0^\infty Pr[X > t]dt.$$

The Appendix contains the formulae and their derivations for Pr[$X > t$] and E[X], given a hypoexponential distribution for the round trip latency. They are required in our analysis later.

In our model of redundant transmissions in a 2-net, a sender node is connected to a receiver node by two networks with independent failure probabilities. For each packet to be transmitted, two copies of it are created and routed independently through the two networks to the destination node. As soon as the destination node gets the first arrival, an acknowledgement is sent back to the source through the network from which the copy is received.† As argued in Section 1, we assume that the extra load incurred by duplicate transmissions does not change the response time of the underlying networks or processors significantly. We also assume that the source has at least two network interface (or control) devices, so that the two copies can essentially be transmitted in parallel.

Let X_M be the conditional random variable representing the effective round trip latency of a *2-net* given that *both copies* of the packet get through the network. We have $X_M = \min\ \{X_1,\ X_2\}$ where X_1 and X_2 are two random variables having the same pdf as X. Moreover,

$$Pr[X_M > t] = Pr[X_1 > t] \cdot Pr[X_2 > t] = \{Pr[X > t]\}^2.$$

So,

$$E[X_M] = \int_0^\infty Pr[X_M > t]dt = \int_0^\infty \{Pr[X > t]\}^2 dt$$

The formula for E[X_M] is a little bit lengthy and is included in the Appendix.

Figure 6 compares the pdfs of X and X_M for $p = 0.95$ ($T_2 = 8.2$), $\lambda_1 = 1$, and $\lambda_2 = 5$. It can be seen from this example that the curve for X_M is more skewed to the left and has a thinner tail. It thus

† A better alternative would be for the destination node to send two acknowledgements back to the source node, riding through both networks, as soon as it gets one copy of the packet. This has the effect of bringing the momentarily slower network up to speed. However, the transmission times would no longer be independent and our analysis would be more complex. Again, this means that 2-nets could perform better than our analysis shows.

has a smaller expected value and a smaller variance.

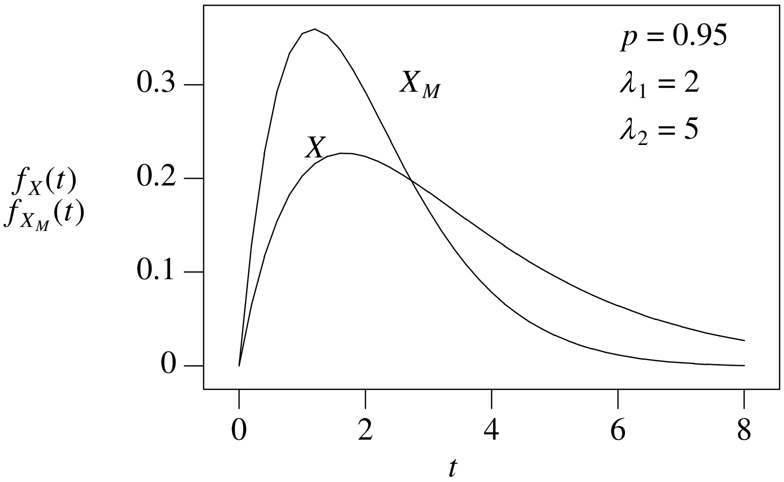


Figure 6 : Comparing X with X_M

Figure 7 shows the variances of X and X_M for $\lambda_1 = 2$ and $\lambda_2 = 5$, and for values of $p > 0.95$. (The Appendix contains the formulae.) Note that in our graph, as well as in others in this paper, we focus on high values of p, i.e., reasonable networks should not lose too many packets.

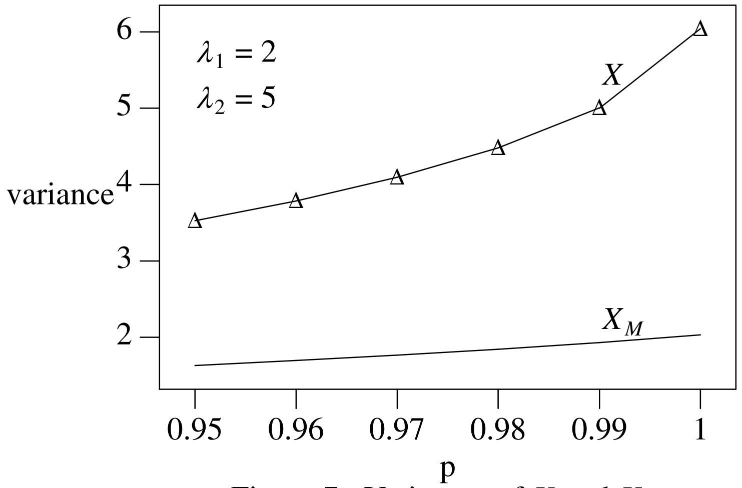


Figure 7 : Variances of X and X_M

Note that the variances increase as p increases. To see why this is so, recall that p is defined by the equation :

$$p = \int_0^{T_2} f(t)dt,$$

where T_2 is the time constant denoting the maximum life time of a packet (see Figure 5). A larger value of p means a bigger T_2. (Given the base transmission time defined by λ_1 and λ_2, to lose fewer packets, we must extend our timeout period.) As T_2 grows, the tail for the round-trip delay distribution grows. This explains why the variances of X and X_M increase as p increases. Moreover, since the distribution of X_M is much skewed to the left (see Figure 6), the area under the tail of the distribution is much thinner and therefore the increase in variance due to a larger p value is much milder than that for X.

In Figures 6 and 7 we assume that packets are not lost. However, we already see the advantages of redundant transmissions: significantly reduced delays and variances. In the following sections we analyze the impact of lost packets and the flow control protocol.

3. A Single Packet Message

Our response time analysis is dependant on the flow control and re-transmission policies used by the networks. This in turn depends on the type of traffic: different protocols are typically used for short messages (e.g., datagram service) as opposed to long messages. Thus, we start our analysis by considering the transmission of a one-packet message.

One-packet transmissions are of special interest because in a large class of distributed algorithms small control messages are exchanged (e.g., in commit or election protocols). In these algorithms there is little opportunity for pipelining data (as in the window protocols we study in Section 5). Essentially, each packet must be sent and acknowledged before the next one can be sent. As we argued in Section 1, the major factor in the running time of these distributed algorithms is the transmission time of these short messages. Thus, the speedup computed by our analysis will essentially be the speedup obtained by these algorithms.

In Section 1 we stated that there are two main ways to use aggressively a 2-net: redundant transmissions and "stream splitting" (sending some of the packets over one net, the rest over the other). In the case of a single packet transmission, stream splitting does not make sense. Thus, we only study redundant transmissions, and compare them to transmissions over a single network.

To send a single packet reliably over a single net, the sender must transmit it and wait for an acknowledgement. If the packet is lost or discarded, the sender eventually times out and sends the packet again. Let R_{one} be the random variable representing the time to successful transmission of one packet in a 1-net. Say a particular packet is transmitted for the first time at time t_1. If there are losses, the packet will be re-transmitted, and eventually an acknowledgement arrives at another time t_2. Then R_{one} will be $t_2 - t_1$. We note that with probability p, we have a successful transmission on the first attempt and therefore $R_{one} = X$, where X is the conditional random variable denoting the round trip latency of a network given that no packet is lost (see Section 2). In general, if the transmission of a packet fails i times before the last trial succeeds, then the response time (R_{one}) is equal to $iT_2 + X$. The first term corresponds to the i timeout intervals and the second term represents the round trip delay caused by the last successful transmission. Hence,

$$Pr[R_{one} = iT_2 + X] = p(1-p)^i \quad i = 0, 1, \ldots$$

This gives us the following expected value of R_{one}.

$$E[R_{one}] = \sum_{i=0}^{\infty} p(1-p)^i [iT_2 + E[X]]$$

$$= E[X] + \frac{T_2(1-p)}{p} .$$

The first term is the expected round trip delay of a successful transmission while the second term is the delay due to an expected number of timeouts.

The probability that at least one copy of a packet gets through a 2-net (p_2) is given by

$$p_2 = 1 - (1-p)^2.$$

If we let Y denote the effective round-trip latency of a 2-net given that *at least one copy* of a packet arrives at the receiver, we have,

$$Y = \frac{p^2}{p_2} X_M + \frac{2p(1-p)}{p_2} X \qquad \text{and,}$$

$$E[Y] = \frac{p^2}{p_2} E[X_M] + \frac{2p(1-p)}{p_2} E[X].$$

Therefore, if $R_{one,M}$ is the random variable representing the time to successful transmission of one packet using a 2-net, we have,

$$E[R_{one,M}] = E[Y] + \frac{T_2(1-p_2)}{p_2},$$

Let us define the speedup (su_{one}) of response time for the one packet case by

$$su_{one} = \frac{\text{expected response time using a 1-net}}{\text{expected response time using a 2-net}}.$$

Therefore,

$$su_{one} = \frac{E[R_{one}]}{E[R_{one,M}]} = \frac{E[X] + \dfrac{T_2(1-p)}{p}}{E[Y] + \dfrac{T_2(1-p_2)}{p_2}}.$$

Figure 8 shows how speedup varies against p, the reliability of the network, for various λ_1's and λ_2's.

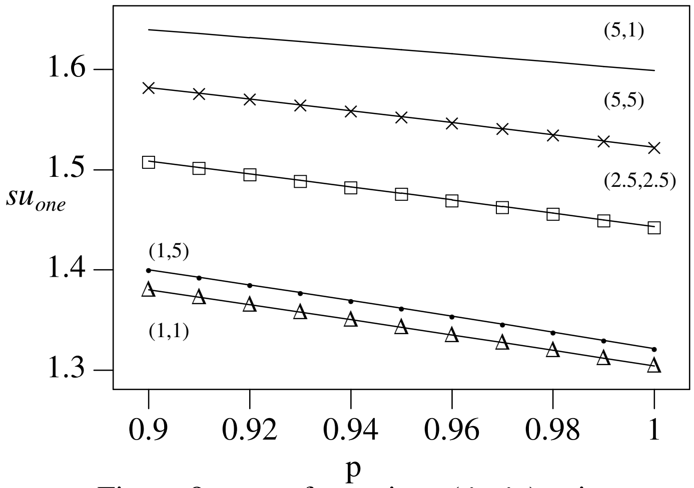


Figure 8 : su_{one} for various (λ_1, λ_2) pairs

We note that a 2-net gives us a smaller expected round trip delay (E[Y] < E[X]) and a smaller number of re-transmissions ($\frac{1-p_2}{p_2} < \frac{1-p}{p}$). Both of these factors contribute to a smaller response time. Taking $p = 0.95$, $\lambda_1 = 5$, $\lambda_2 = 1$ for example, E[X] = 6.04, E[Y] = 4.24, $T_2(1 - p)/p = 0.89$, and $T_2(1 - p_2)/p_2 = 0.042$. The ratio $\dfrac{E[Y] - E[X]}{T_2[(1 - p)/p - (1 - p_2)/p_2]}$ measures the relative contribution to response time speedup by the 2 factors. For our example values of p, λ_1, and λ_2, this ratio is equal to 1.804/0.848 = 2.13. This means that the reduction in expected round trip delay contribute more than twice as much as the reduced number of re-transmissions. That is, the round trip reduction is the more important factor. We also notice that as the value of p increases, the difference between p and p_2 decreases, while the difference between E[Y] and E[X] increases (see the formula of E[Y]). Therefore, for reasonably large values of p (e.g., $p > 0.95$), a large portion of the speedup is due to the reduced variability of the round trip latency of the network.

To illustrate the effect of λ_1 on speedup,† we choose two representative values for p (0.95) and λ_2 (3), and plot speedup as a function of λ_1 in Figure 9. The curve shows us that the more unpredictable the round-trip delay (bigger λ_1) is, the more speedup advantage we can obtain by using a 2-net. The conclusion drawn from Figures 7 and 8 is that the speedups due to redundant transmission can be significant, anywhere from 20 to 65 percent. These results are obtained over a wide range of base distributions and network reliabilities.

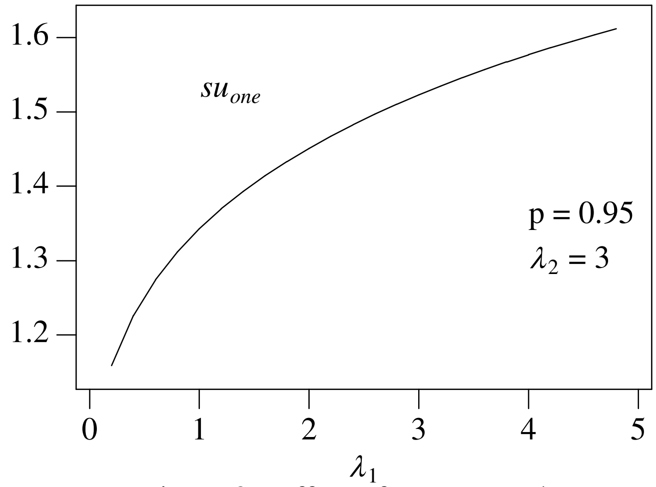


Figure 9 : Effect of λ_1 on speed up

† Recall that λ_1 is a parameter controlling the shape of the round trip delay distribution of a component network — the larger λ_1 is, the more spread out the distribution is (see Figure 4).

4. Medium Sized Message

In this section, we study the transmission of a medium sized message over a high latency network. A medium sized message consists of more than one packet but is not large enough to fill up the network pipeline. By this we mean that the round trip latency of the network is larger than the total time taken by the sender to pump all packets onto the network. If we let T_a be the amount of time taken by the sender to put a packet onto the network, then with a medium sized message of k packets, we have,

$$kT_a < \Delta^{\S}$$

The transmission protocol we are going to study is very simple. When the sender sends out a packet, p_i, a timer for p_i is enabled. When the receiver receives a packet p_i, an acknowledgement ACK_i is sent back to the sender. If the sender does not receive an ACK_i T_2 time units after p_i is sent, p_i is retransmitted. We assume that k is sufficiently small that each packet can be individually buffered at the receiver and can be individually acknowledged. We choose to study this *selective repeat* protocol, even though it requires more complicated logic and more buffer space at the receiver than the more commonly used go-back-N algorithm [22], because it is more efficient and the additional buffer space requirement is easily accommodated for small k. Moreover, go-back-N is more sensitive to p for the range we are interested $(0.9 < p < 1.0)$[16]. This implies that an improvement in error rate increases network utilization by a bigger margin when go-back-N is used than when selective repeat is used. As we will see later in this section, a main advantage in using a 2-net is an improved error rate; therefore, we expect speedups that are greater than what we show in this section for go-back-N using 2-net.

4.1 Computing Response Time for a Single Network

In order to demonstrate how response time is calculated, we consider the following two scenarios for transmission using 1-net.

---

§ Δ is the minimum round trip latency of a network. See Section 2.

Scenario (1) : No Retransmission

In this case all packets survive through the first transmission. Therefore,

$$\text{Response time} = kT_a + E[X],$$

where kT_a is the time taken for the source to put all k packets onto the network and E[X] is the expected time taken for the last packet (p_k) to get to the destination and its acknowledgement to be received by the source (see Figure 10).

Scenario (2) : One Retransmission

In this case, some packets get lost in the first trial, but make their ways through the second time. To illustrate, suppose we want to transmit 7 packets at time 0 (see Figure 11). In our first trial packets #3 and #5 are lost. At time $T_2 + 2T_a$ and time $T_2 + 4T_a$ respectively, the timers of p_3 and p_5 expire and the two lost packets are retransmitted. Finally, at time $T_2 + 5T_a + \mathrm{E}[X]$, ACK_5 is received and the transmission is completed. Note that the total transmission time is not affected by the fact that p_3 gets lost in its first trial.

We can generalize the analysis to the the case when n retransmissions are required (i.e., there is at least one packet that takes $n+1$ trials to complete). Let p_j be the last packet received at the destination in the last $(n+1^{st})$ trial. Then the response time for sending the message would be $nT_2 + E[X] + jT_a$ (see Figure 12). The probablity of such event happening is :

$$Pr[\text{Response time} = nT_2 + E[X] + jT_a] = Pr[A \ \& \ B \ \& \ C]$$

where,

A = the event that packet p_j is received at the $n + 1^{st}$ trial.

B = the event that none of the transmissions of $p_1, \ldots, p_{j-1}$ takes more than $n + 1$ trials.

C = the event that none of the transmissions of $p_{j+1}, \ldots, p_k$ takes more than n trials.

Letting $q = 1 - p$, it can be seen that

$\Pr(A) = q^n p$,

$\Pr(B) = (1 - q^{n+1})^{j-1}$, and

$\Pr(C) = (1 - q^n)^{k-j}$.

Therefore,

$$Pr[\text{Response time} = nT_2 + E[X] + jT_a]$$

$$= Pr(A) \cdot Pr(B) \cdot Pr(C)$$

$$= q^n p (1 - q^{n+1})^{j-1} (1 - q^n)^{k-j}.$$

Let R_1 be the expected response time of transmitting a medium sized message of k packets using a 1-net. Then,

$$R_1 = \sum_{n=0}^{\infty} \sum_{j=1}^{k} q^n p (1 - q^{n+1})^{j-1} (1 - q^n)^{k-j} (nT_2 + E[X] + jT_a).$$

As shown in the Appendix, this can be rewritten as

$$R_1 = E[X] + B(q,k)T_2 + C(q,k)T_a,$$

where,

$$B(q,k) = \sum_{i=1}^{k} \binom{k}{i} (-1)^{i+1} \frac{q^i}{1 - q^i} .$$

$$C(q,k) = \sum_{j=1}^{k} pj \sum_{i_1=0}^{j-1} \sum_{i_2=0}^{k-j} \binom{j-1}{i_1} \binom{k-j}{i_2} (-1)^{i_1+i_2} \frac{q^{i_1}}{1 - q^{i_1+i_2+1}} .$$

4.2 Redundant Transmissions

By duplicating packets and transmitting the duplicates over a second network in parallel with the originals, we effectively form a 2-net that has a shortened expected round trip latency (E[Y]) and a higher effective probability of successful transmission. The probability of a successful transmission is $p_2 = 1 - (1 - p)^2$, and the probability of a loss is $1 - p_2 = 1 - (1 - (1 - p)^{2)} = q^2$. Hence, if we let $R_{2,red}$ be the response time of a 2-net using redundant transmission, we have,

$$R_{2,red} = E[Y] + B(q^2,k)T_2 + C(q^2,k)T_{a.}$$

In this expression, $B(q^2,k)T_2$ is likely to be the dominating factor. This is because for a high latency network, T_2, the timeout interval, should be much larger than T_a, the packet duration. Constant T_2 should also be larger than $E[X]$, otherwise the network will time out too early. Thus, before studying $R_{2,red}$ we study the factor $B(q^2,k)$. (Note also that value $E[Y]$ has been studied in Section 2 and Figure 6.)

Intuitively, $B()$ is the expected number of retransmissions. Figure 13 shows $B(q,k)$ as a function of p (= $1 - q$) for various values of k. From the figure, we see that as the network becomes more and more reliable (as p increases), the number of retransmission ($B()$) decreases. Moreover, the slope of the curves gets steeper as k becomes larger. Therefore, as the size of the message increases, the improvement in response time attributable to a higher probability of successful transmission (p) also increases.

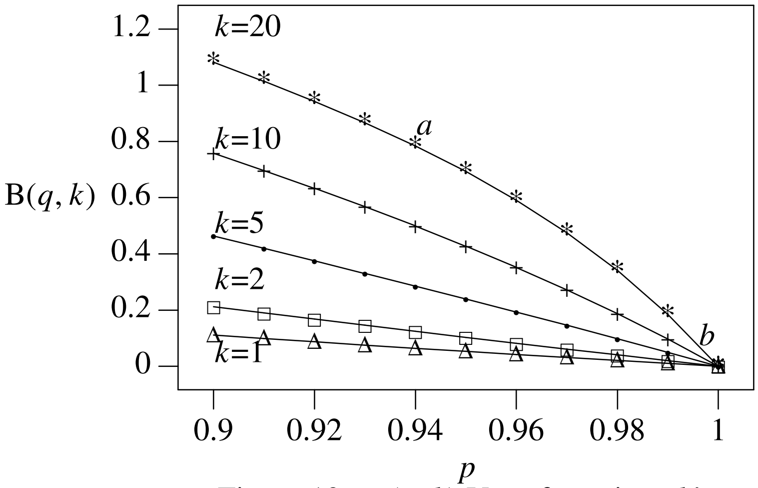
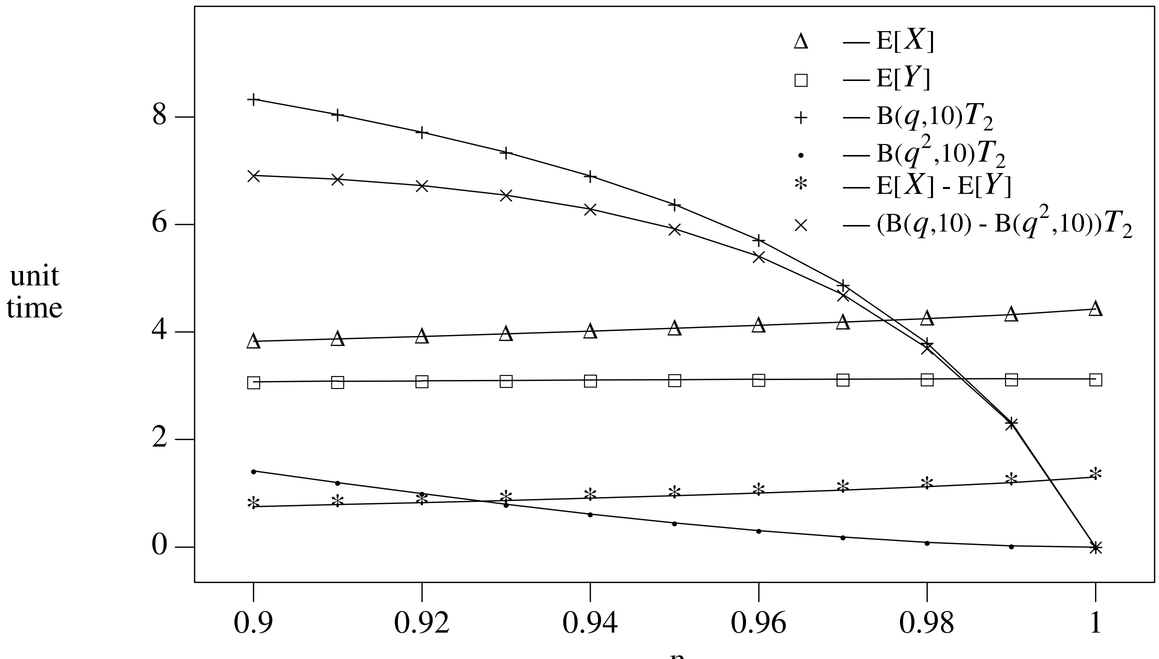


Figure 13 : $B(q,k)$ Vs p for vaious k's

Redundant transmission helps reduce retransmissions by essentially increasing the effective value of p (to $p_2 = 1 - (1-p)^2$). For example, if $p = 0.94$, $k = 20$, the expeccted number of retransmissions is about 0.79 (point a in Figure 13). A 2-net pushes p to p_2 (= 0.9964), and $B(0.0036, 20) \approx$ 0.06 (point b).

Figure 14 shows $B(q^2, 10)T_2$, $B(q, 10)T_2$ and their difference for $\lambda_1 = 2$, $\lambda_2 = 5$, and for various values of p. The value of $T_2(B(q,10) - B(q^2,10))$ represents the improvement, due to reduced retransmissions, brought about by a 2-net.



The figure also shows $E[X]$, $E[Y]$, and $E[X] - E[Y]$. The value $E[X] - E[Y]$ represents the response time improvement provided by a 2-net due to reduced transmit times. Comparing $T_2(B(q, 10) - B(q^2, 10)))$ and $E[X] - E[Y]$, we see that the savings on retransmission contributes to a more significant factor in speedup, unless the underlying network is very reliable.

Now that we have studied the factors that make up the transmission time, let us study the overall speedup obtainable by a 2-net. Define $su_{k,red}$ to be the speedup obtainable by a 2-net in sending a k-packet message using redundant transmission. We have,

$$su_{k,red} = \frac{\text{response time of 1-net}}{\text{response time of 2-net}}$$

$$= \frac{E[X] + B(q,k)T_2 + C(q,k)T_a}{E[Y] + B(q^2,k)T_2 + C(q^2,k)T_a} .$$

For our evaluation we need to select a value for T_a. Typically, for a high latency network, T_a varies from 1/30 (satellite link) to 1/100 (optical fiber, long-distance transmission) of Δ, the total latency[1]. For our next graph we choose a medium value of $T_a = \Delta/50$. Since we are assuming that $kT_a < \Delta$ (medium size messages), we will let k range from 2 to 50. Results are plotted in Figure 15 which shows $su_{k,red}$ as a function of p.

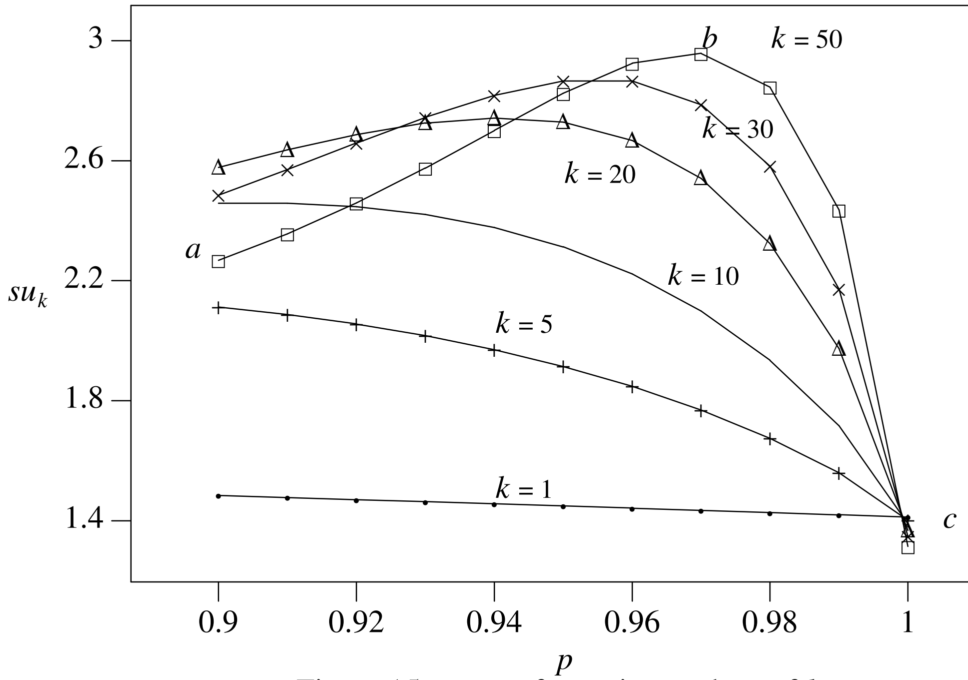


Figure 15 : $su_{k,red}$ for various values of k.

Taking a look at the $su_{k,red}$ expression, we see that for small values of p, the number of retransmissions is large, and the $B()$ terms tend to dominate both in the numerator and in the denominator. And hence $su_{k,red}$ behaves more or less like $B(q,k)/B(q^2,k)$. $B(q^2,k)$ decreases faster than $B(q,k)$ as q decreases, we would expect $su_{k,red}$ to increase as p increases for small values of p. This explains, for example, the part of the curve labeled *ab* in Figure 15.

Now as p increases further, the number of retransmissions, (i.e. $B()$) becomes smaller, and $su_{k,red}$ therefore, tends to (E[X]+Δ)/(E[Y]+Δ), a fixed value. This explains the curves labeled *bc* in the graph.

Lastly, as more packets are sent (k increases), more retransmissions are expected, and this leads to bigger $B()$ terms. Therefore, the $B()$ terms tend to donimate for a wider range of p values. This explains why the curves for larger k peak more to the right.

4.3 Stream Splitting

With stream splitting, we divide the set of k packets into two equal sized groups and send them independently over two networks. If we let TT_A, and TT_B be the random variables denoting the response times of transmitting $k/2$ packets over each network, then,

$$E[TT_A] = E[TT_B] = E[X] + B(q,k/2)T_2 + C(q,k/2)T_a.$$

The total transmission time $R_{2,split}$ would be

$$\begin{aligned} R_{2,split} &= \max\ \{TT_A, TT_B\} \\ &\geq TT_A. \end{aligned}$$

Therefore, the expected value of the response time using stream splitting $R_{2,split}$ has the following lower bound :

$$R_{2,split} \geq E[X] + B(q,k/2)T_2 + C(q,k/2)T_a.$$

Hence, an upper bound on the speedup $su_{k,split}$ can be computed as

$$su_{k,split} \leq \frac{R_1}{E[TT_A]}$$

$$\leq \frac{E[X] + B(q,k)T_2 + C(q,k)T_a}{E[X] + B(q,k/2)T_2 + C(q,k/2)T_a} .$$

Figure 16 shows this upper bound as a function of p for various values of k.

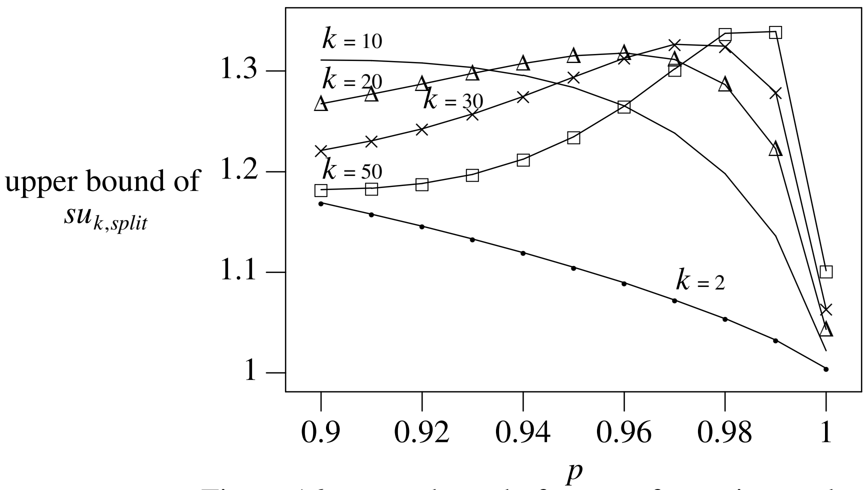


Figure 16 : upper bound of $su_{k,split}$ for various values of k.

We observe that the maximum speedup obtainable using stream splitting in this case can hardly exceed 35% while at least 40% improvement in response time is almost always guaranteed by redundant transmission (see Figure 15). Therefore, we conclude that redundant transmission is a better choice for transmitting medium sized messages.

By generating graphs like Figure 15, intelligent decision can be made concerning the choice of transmision mode. Say for example, we have two slow but cheap networks, and a fast and expensive network, and we have a soft real-time application. By considering the costs of transmitting the message, the size of the message, and the penalty of a missed deadline, we can choose to send the message through (1) the slow network, or (2) the polynet formed by the slow networks, or (3) the fast network. Moreover, we can also consider the level of reliability, and the variability in transmission time (a measure of risk) each transmission mode provides.

5. A Long Message

In this section, we analyze the transmission of a long message (many packets) over a high latency network. For such transmissions, a sliding window protocol is typically used. In particular, we analyze protocol #6 (selective repeat) in [22]. This is the most complete sliding window algorithm presented in that textbook. Our results are also applicable to other algorithms of this type with slight modification.

Let m, the message to be sent consist of k packets $p_1, p_2, \ldots, p_k$. Instead of having the source node block every time it sends out a packet and wait for the acknowledgement, we allow the sender to transmit packets continuously. When the sender puts a packet p_i onto the network, a timer for it is started. If acknowledgements always arrive before timeout, the sender just keeps on sending one packet after another, without delays. However, if no acknowledgement for a packet p_i is received when the timer goes off, p_i is re-transmitted. The receiver transmits an acknowledgement A_i only if $p_1, \ldots, p_i$ have all been received. We assume that the receiver has a lot of buffer space (a large window), so packets received out of sequence are buffered (but not acknowledged). Since we are interested in networks with reasonable reliability ($p \geq 0.95$), we make the following simplifying assumption: the probability that the transmission of a given packet fails twice in a row is extremely small. As a result, we assume that if a packet is re-transmitted, it is properly received and acknowledged. Once again, the effect is that we underestimate (by a small amount) the potential speedup generated by a 2-net.

If we let T_a be the amount of time taken by the sender to put a packet onto the network, then we can think of time being divided into slots, each slot being T_a units long. If the network is perfect, no packets will ever get lost, and as have been explained in last section,

Response time = $kT_a + E[X]$,

Complications arise when the network is not perfect. We let l be the timeout interval in units of time slots (i.e., $l = T_2/T_a$), and let r be the round trip latency in units of time slots (i.e., $r = E[X]/T_a$). Consider the following sample scenario : packets $p_1, \ldots, p_{i-1}$ get through the network but not p_i. So at time slot number $i + l$, the timer for p_i goes off and p_i is re-transmitted. The sequence

$$\cdots\ p_i\ p_{i+1}\ \cdots\ p_{i+l}\ p_i$$

$$\longrightarrow \text{time}$$

shows the stream of packets sent by the source node.

In sliding window protocols, typically packets $p_{i+1} p_{i+2} \cdots$ cannot be acknowledged by the destination until packet p_i is received. (The acknowledgement for p_{i+1} implicitly acknowledges all packets through $i + 1$.) Therefore, at the source the timers for packets $p_{i+1,} p_{i+2}$ .. etc. will also go off successively and $p_{i+1,} p_{i+2}$... etc. will be re-transmitted. Thus, the sequence of packets is :

$$\cdots\ p_i\ p_{i+1}\ \cdots\ p_{i+l}\ p_i\ p_{i+1}\ p_{i+2}\ \cdots$$

As mentioned earlier, we assume that the second trial of the transmission of p_i succeeds. So at time slot number $i + l + 1 + r$, the sender gets the acknowledgement for p_{i+l} and can proceed with the next packet, p_{i+l+1}.

$$\cdots\ p_i\ p_{i+1}\ \cdots\ p_{i+l}\ p_i\ \cdots\ p_{i+r}\ p_{i+l+1}$$

Note that missing a single packet, p_i, not only caused us to waste one slot re-transmitting, but also caused the unnecessary transmission of r other packets (which were not lost, but simply unacknowledged). Thus, the number of slot overhead (d) that can be charged to the loss of p_i is equal to $d = (l + r + 1) - (l) = r + 1$.

Since $r = E[X]/T_a$ (defined above), where E[X] is the expected round-trip latency of a network, then,

$$d = r + 1 = E[X]/T_a + 1.$$

We note that if p_{i+1} is also lost in the first trial but makes its way through the second time, no additional overhead is incurred by its loss. The number of wasted slots is still d. In general, the loss of a packet p_i causes a "burp," an overhead of d additional time slots. However, any subsequent losses of $p_{i+1}, \ldots, p_{i+r}$ do not cause any additional "burps."

We compute the overhead in the general case as follows. Let N_i be the expected total number of burps that occur given that packets $p_1, \ldots, p_{i-1}$ have arrived safely ($p_{1,} \ldots, p_{i-1}$ caused no burps) at the destination. If packet p_i arrives intact, the number of burps is be equal to N_{i+1}. Conversely, if packet p_i is lost, one burp occurs and this burp is good for any subsequent losses of an additional $d-1$ packets. The expected number of burps in this case is thus $1 + N_{i+d}$. We can formalize the above idea by the following recurrence :

$$\begin{array}{ll} N_i = pN_{i+1} + (1-p)[1 + N_{i+d}] & i \le k, \\ N_i = 0 & i > k. \end{array}$$

This recurrence enables us to compute N_1, the expected number of burps in transmitting the k-packet long message.

The total transmission time (TT) is given by

$$\begin{aligned} TT &= T_a(k + N_1 d) + E[X] \\ &= kT_a(1 + N_1 d/k) + E[X]. \end{aligned}$$

The above formula is useful in the following subsections for computing speedups.

5.1 Redundant Transmissions

Let us now analyze the case where the full stream of packets is sent over two networks. The analysis is similar, except that we have a different expected round trip latency, E[Y], and therefore a different burp length d', which is equal to E[Y]/T_a + 1. Let the sequence $\{N'_i\}$ be determined by the following recurrence

$$\begin{array}{ll} N'_i = pN'_{i+1} + (1-p)[1 + N'_{i+d'}] & i \le k, \\ N'_i = 0 & i > k. \end{array}$$

Then the speedup of a 2-net (with redundancy) over a conventional single network ($su_{1n/red}$) would be

$$su_{1n/red} = \frac{kT_a(1 + N_1 d/k) + E[X]}{kT_a(1 + N'_1 d'/k) + E[Y]}\,.$$

When k is large (e.g., we want to transfer a big file which consists of hundreds or thousands of packets), E[X] and E[Y] are negligible compared with the other terms. Hence,

$$su_{1n/red} \simeq \frac{kT_a(1 + N_1 d/k)}{kT_a(1 + N_1' d'/k)} \quad \text{large } k.$$

Defining $n = \lim_{k \to \infty} N_1/k$, and $n' = \lim_{k \to \infty} N_1'/k$, we have

$$\lim_{k \to \infty} su_{1n/red} = \frac{1 + nd}{1 + n'd'} .$$

In the Appendix we show that $n = q/[1 + (d - 1)q]$ and $n' = q'/[1 + (d' - 1)q']$, where $q = 1 - p$ and $q' = 1 - p_2$ (and as before, $p_2 = 1 - (1 - p)^2$). Figure 17 shows $su_{1n/red}$ plotted against p, for various values of λ_1 and λ_2.

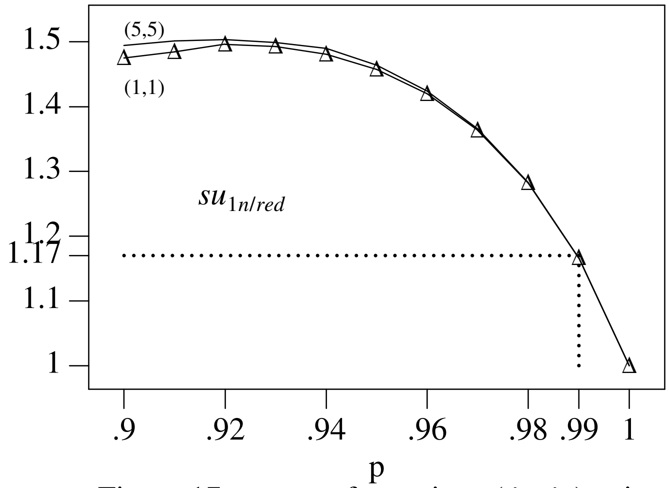


Figure 17 : $su_{1n/red}$ for various (λ_1,λ_2) pairs

We observe that if the network never loses packets (p = 1), the number of burps that occur is zero and a 2-net has no advantage over a 1-net (speedup is one). However, speedup rises sharply when p starts decreasing from 1. For example, even if the network drops only one packet out of a hundred, a 2-net can already achieve a speedup of 17%. This is due to the magnification effect of burps: a single loss causes a break in the pipeline and a string of re-transmissions. If packet delivery rate drops further, to $p = 0.9$, a gain of 50 percent is obtainable with a 2-net.

The curves shown in Figure 17 correspond to networks with widely differing round-trip variabilities (see Figure 4). The overlapping of their speedup curves shown in the figure suggests that the speedup in the transmission of large messages is quite immune to the network transmission delay distribution.

Finally, we note that it is possible to modify sliding window protocols to avoid the burps we have analyzed. The key is to have the destination acknowledge packets individually. That is, the acknowledgement for p_i now means that p_i was received, but it says nothing about p_{i-1} or the previous packets. This generates slightly more acknowledgement traffic.† On the other hand, the number of wasted slots per loss is now simply one. With p being the probability of a successful transmission, the expected number of transmissions required to get a packet from the source node to the destination node is now $1/p$. Therefore, the total transmission time is kT_a/p for a 1-net, and kT_a/p_2 for a 2-net. The speedup made possible by a 2-net is thus equal to

$$\frac{p_2}{p} = \frac{[1-(1-p)^2]}{p} = 2-p.$$

For $p > 0.95$, a 2-net can only buy us a 5% speedup, probably not worth the extra effort spent.

5.2 Stream Splitting

A second alternative is to split the stream into two streams, each with $k/2$ packets. Thus, packet p_1 is sent over one network, packet p_2 over the second one, p_3 over the first, p_4 over the second, and so on. This approach is only feasible if there is enough buffer space at the source and destination to keep two full windows.

Consider a 2-net that is composed of two networks A and B. Let TT_A and TT_B be the random variables denoting the response times of transmitting $k/2$ packets over networks A and B respectively. Then, for large k,

$$E[TT_A] = E[TT_B] = (k/2)T_a(1+nd).$$

The total transmission time (TT) of the 2-net would be

† If packets arrive in order, then the modified protocol uses the same number of acknowledgements as the conventional protocol: one per received packet. It is only when packets arrive out of order that the modified protocol would send more acknowledgements.

$$TT = \max \{ TT_A, TT_B \}.$$

In the Appendix we show that, $\lim_{k \to \infty} \sigma_{TT_A} / E[TT_A] = 0$ where σ_{TT_A} is the standard deviation of TT_A (A similar statement holds for TT_B). Therefore, for large value of k, the transmission time of a network will not deviate from its mean by any significant amount. Hence,

$$E[TT] = E[TT_A] = (k/2) T_a (1 + nd) \quad \text{large } k.$$

Comparing this to the response time of a 1-net and to the response time of a 2-net with redundant transmission, we get

$$su_{1n/split} \simeq 2$$

$$su_{red/split} \simeq \frac{2(1 + n'd')}{1 + nd}.$$

The fact that stream splitting provides a speedup of 2 over a single network is what one would intuitively expect, given that twice the bandwidth is available. Redundant transmissions also provide a speedup with respect to a 1-net (Figure 17), but our $su_{red/split}$ shows that splitting is better. Figure 18 shows $su_{red/split}$ as a function of p. As can be seen, splitting is superior by a factor ranging from 1.33 (at $p = 0.9$) to 2 (at $p = 1$).

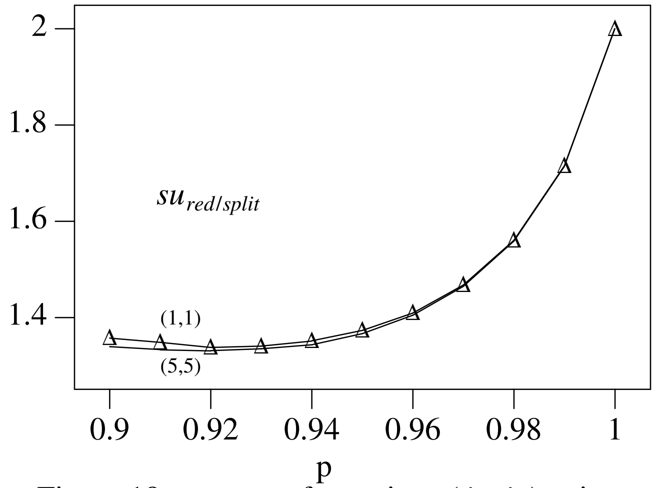


Figure 18 : $su_{red/split}$ for various (λ_1, λ_2) pairs

Note that redundant transmissions could be more effective than splitting at lower values of p. However, we do not consider values of p less than 0.9 realistic (plus our analysis is only for relatively reliable networks). Also note that stream splitting requires two full sets of windows, so it may be infeasible if memory is tight. In conclusion, for "moderately reliable" networks where memory is plentiful, stream splitting is the best alternative for very long messages.

6. Conclusions

If two computers wish to communicate, it is much more efficient to have a single path linking them, rather than two separate paths, each with half the bandwidth[11]. However, all reliable systems opt for the two path solution to avoid vulnerability to a single failure. So, given that separate paths exist, it is possible to use aggressive transmission techniques to improve the response time (and decrease the variance) of critical messages.

For sending a single short packet, duplicate transmissions may improve response time by 20 to 60 percent, depending on the network characteristics. This may in turn improve the overall performance of distributed algorithms that are limited by network latency (not by network throughput). In turn these savings may reduce resource contention, as is the case for distributed commit protocols.

We have also analyzed an intermediate case, in which a small number of packets is to be sent over a high latency network. Here again, duplication pays off, as it is better to be pessimistic and send copies, rather than to wait and see if all the packets made it. As a matter of fact, since there is now more than a single packet, the probability that at least one is lost or delayed increases, making the payoff for duplication even larger than for the single packet case. (Stream splitting is not effective, as latency and not bandwidth is the major factor)

For sending long messages (e.g., file transfer), full pipelining makes speedup rather insensitive to network latency characteristics. In this case, splitting the stream over 2 networks is superior, by up to a factor of 2, to redundant transmission. This is true even for unreliable networks.

Our last observation concerns the price and speed of communication lines. Sometimes vendors provide lines of varying qualities and prices. (A higher grade line needs more expensive equipment, more repeaters, needs to run slower, and so on.) Given that multiple paths exist and that aggressive transmissions are used, it may be feasible to use *lower* grade lines (either to save money or to get faster lines). That is, the duplicate paths compensate for losses that may occur on one path, so good performance can still be obtained. In other words, aggressive transmissions and redundant paths open the door for many interesting alternatives in the price/performance/reliability spectrum.

Appendix

Computing Pr[X>t], E[X] and Var[X]

$$Pr[X > t] = \int_t^\infty h(t)dt.$$

For $\Delta \le t \le T_2$, we have

$$Pr[X > t] = \int_t^{T_2} h(t)dt$$

$$= \frac{1}{p}\int_t^{T_2} f(t)dt$$

$$= \frac{1}{p}[\int_t^\infty f(t)dt - \int_{T_2}^\infty f(t)dt]$$

$$= \frac{1}{p}[\int_t^\infty \frac{\lambda_1+\lambda_2}{\lambda_1^2}[e^{-(t-\Delta)/\lambda_1} - e^{-(t-\Delta)(\frac{\lambda_1+\lambda_2}{\lambda_1\lambda_2})}]dt - (1-p)]$$

$$= \frac{1}{p}[\frac{\lambda_1+\lambda_2}{\lambda_1}e^{-(t-\Delta)/\lambda_1} - \frac{\lambda_2}{\lambda_1}e^{-(t-\Delta)(\frac{\lambda_1+\lambda_2}{\lambda_1\lambda_2})} - (1-p)].$$

We also have

$$E[X] = \int_0^\infty Pr[X > t]dt$$

$$= \int_\Delta^{T_2} Pr[X > t]dt + \int_0^\Delta 1 \cdot dt$$

$$= \frac{1}{p}[(\lambda_1+\lambda_2)\alpha_1 - \frac{\lambda_2^2}{\lambda_1+\lambda_2}\alpha_2 - (T_2-\Delta)(1-p)] + \Delta,$$

where

$$\alpha_1 = 1 - e^{(\frac{-(T_2-\Delta)}{\lambda_1})},$$

$$\alpha_2 = 1 - e^{-(T_2-\Delta)(\frac{\lambda_1+\lambda_2}{\lambda_1\lambda_2})}.$$

Moreover,

$$Var[X] = E[X^2] - E^2[X]$$

$$= \int_{\Delta}^{T_2} t^2 h(t)dt - E^2[X].$$

This equation was solved numerically using Mathematica to obtain Figure 7.

Formula for E[X_M] and Var[X_M]

$$E[X_M] = \int_0^{\infty} Pr[X_M > t]dt = \int_0^{\infty} \{Pr[X > t]\}^2 dt$$

$$= \frac{1}{p^2}\,[\zeta_1\beta_1 + \zeta_2\beta_2 + \zeta_3\beta_3 + \zeta_4\beta_4 + \zeta_5\beta_5 + (1-p)^2(T_2-\Delta)] + \Delta$$

where

$$\zeta_1 = \frac{(\lambda_1+\lambda_2)^2}{2\lambda_1}, \qquad \beta_1 = 1 - exp(\frac{-2(T_2-\Delta)}{\lambda_1}),$$

$$\zeta_2 = \frac{-2(\lambda_1+\lambda_2)\lambda_2^2}{(\lambda_1+2\lambda_2)\lambda_1}, \qquad \beta_2 = 1 - exp(-(T_2-\Delta)(\frac{\lambda_1+2\lambda_2}{\lambda_1\lambda_2})),$$

$$\zeta_3 = \frac{\lambda_2^3}{2\lambda_1(\lambda_1+\lambda_2)}, \qquad \beta_3 = 1 - exp(-2(T_2-\Delta)(\frac{\lambda_1+\lambda_2}{\lambda_1\lambda_2})),$$

$$\zeta_4 = -2(1-p)(\lambda_1+\lambda_2), \qquad \beta_4 = 1 - exp(\frac{-(T_2-\Delta)}{\lambda_1}),$$

$$\zeta_5 = \frac{2(1-p)\lambda_2^2}{\lambda_1+\lambda_2}, \qquad \beta_5 = 1 - exp(-(T_2-\Delta)(\frac{\lambda_1+\lambda_2}{\lambda_1\lambda_2})).$$

Letting $h_M(t)$ be the pdf of X_M, then

$$h_M(t) = \frac{d}{dt}\,(Pr[X_M \le t])$$

$$= \frac{d}{dt}(1 - Pr[X_M > t]).$$

Therefore,

$$Var[X_M] = E[X_M^2] - E[X_M]$$

$$= \int_{\Delta}^{T_2} t^2 h_M(t)dt - E[X_M].$$

This last equation was also solved numerically.

Formula for R_1

$$R_1 = \sum_{n=0}^{\infty} \sum_{j=1}^{k} q^n p(1 - q^{n+1})^{j-1}(1 - q^n)^{k-j}(nT_2 + E[X] + jT_a),$$

which can be broken down into three terms as follows :

$$R_1 = A(q,k) \cdot E[X] + B(q,k) \cdot T_2 + C(q,k) \cdot T_a$$

where,

$$A(q,k) = \sum_{n=0}^{\infty} \sum_{j=1}^{k} q^n p(1 - q^{n+1})^{j-1}(1 - q^n)^{k-j},$$

$$B(q,k) = \sum_{n=0}^{\infty} \sum_{j=1}^{k} q^n p(1 - q^{n+1})^{j-1}(1 - q^n)^{k-j} n, \text{ and}$$

$$C(q,k) = \sum_{n=0}^{\infty} \sum_{j=1}^{k} q^n p(1 - q^{n+1})^{j-1}(1 - q^n)^{k-j} j.$$

Now, $A(q,k)$ is the sum of the probabilities of all the cases, therefore, $A(q,k) = 1$.

Let

$$D(q,n,k) = \sum_{j=1}^{k} q^n p(1 - q^{n+1})^{j-1}(1 - q^n)^{k-j},$$

then,

$$B(q,k) = \sum_{n=0}^{\infty} nD(q,n,k).$$

Now, the probability that a packet gets through the network within n trials is $(1-q^n)$. The probability that all k packets get through the network within n trails is therefore equal to $(1-q^n)^k$. Since $D(q,n,k)$ is the probability that exactly n retransmissions (i.e., $n+1$ transmissions) are required, $D(q,n,k)$ = Prob[All k packets get through the network within $n+1$ transmissions but not all k packets get through the network within n transmissions]. We thus have,

$$D(q,n,k) = (1-q^{n+1})^k - (1-q^n)^k.$$

Hence,

$$B(q,k) = \sum_{n=0}^{\infty} n[(1-q^{n+1})^k - (1-q^n)^k]$$

$$= \sum_{n=0}^{\infty} n\left[\sum_{i=0}^{k}\binom{k}{i}q^{(n+1)i}(-1)^i - \sum_{i=0}^{k}\binom{k}{i}q^{ni}(-1)^i\right]$$

$$= \sum_{n=0}^{\infty} n \sum_{i=0}^{k}\binom{k}{i}(-1)^i q^{ni}(q^i-1)$$

$$= \sum_{i=1}^{k}\binom{k}{i}(-1)^i(q^i-1)\sum_{n=0}^{\infty} n(q^i)^n$$

$$= \sum_{i=1}^{k}\binom{k}{i}(-1)^i(q^i-1)\,\frac{q^i}{(1-q^i)^2}$$

$$= \sum_{i=1}^{k}\binom{k}{i}(-1)^{i+1}\,\frac{q^i}{1-q^i}\,.$$

Finally,

$$C(q,k) = \sum_{n=0}^{\infty}\sum_{j=1}^{k} q^n p(1-q^{n+1})^{j-1}(1-q^n)^{k-j}\, j,$$

$$= \sum_{j=1}^{k} pj \sum_{n=0}^{\infty} q^n(1-q^{n+1})^{j-1}(1-q^n)^{k-j}$$

$$= \sum_{j=1}^{k} pj \sum_{n=0}^{\infty} q^n \sum_{i_1=0}^{j-1} \binom{j-1}{i_1} (-1)^{i_1} q^{i_1(n+1)} \sum_{i_2=0}^{k-j} \binom{k-j}{i_2} (-1)^{i_2} q^{ni_2}$$

$$= \sum_{j=1}^{k} pj \sum_{i_1=0}^{j-1} \sum_{i_2=0}^{k-j} \binom{j-1}{i_1} \binom{k-j}{i_2} (-1)^{i_1+i_2} \sum_{n=0}^{\infty} q^{n+i_1(n+1)+ni_2}$$

$$= \sum_{j=1}^{k} pj \sum_{i_1=0}^{j-1} \sum_{i_2=0}^{k-j} \binom{j-1}{i_1} \binom{k-j}{i_2} (-1)^{i_1+i_2} \frac{q^{i_1}}{1-q^{i_1+i_2+1}} .$$

Computing n and n'

The original recurrence (for a 1-net) is :

$$\begin{aligned} N_i &= pN_{i+1} + (1-p)[1 + N_{i+d}] \quad && i \le k, \\ N_i &= 0 && i > k. \end{aligned}$$

We would like to compute $n = \lim_{k\to\infty} N_1/k$. But this is equivalent to computing $\lim_{i\to\infty} M_i/i$ for the following recurrence :

$$\begin{aligned} M_i &= (1-q)M_{i-1} + q(M_{i-d} + 1) \quad && \text{for } i \ge 1, \\ M_i &= 0 && \text{for } i \le 0, \end{aligned}$$

where $q = 1 - p$.

We can interpret M_i as the expected number of burps that occur during the transmission of an i-packet long message.

Simple inductions show that for all i :

(1) M_i is a polynomial in q, and

(2) M_i has 0 as its constant term.

We can therefore let $M_i = q \sum_{j=0}^{\infty} a_{i,j} q^j$. For example, since $M_0 = 0$ and $M_1 = q$, we have

$$a_{0,j} = 0 \qquad \text{for all } j, \quad \text{and}$$

$$a_{1,0} = 1, \quad a_{1,j} = 0 \quad j > 0.$$

By the recurrence, we have

$$M_i = (1-q)q\sum_{j=0}^{\infty} a_{i-1,j}q^j + q(q\sum_{j=0}^{\infty} a_{i-d,j}q^j + 1)$$

$$= q[\sum_{j=0}^{\infty} a_{i-1,j}q^j - \sum_{j=0}^{\infty} a_{i-1,j}q^{j+1} + \sum_{j=0}^{\infty} a_{i-d,j}q^{j+1} + 1]$$

$$= q[\sum_{j=1}^{\infty}(a_{i-1,j} - a_{i-1,j-1} + a_{i-d,j-1})q^j + a_{i-1,0} + 1] \quad (1).$$

So, for $i > d$,

$$a_{i,0} = a_{i-1,0} + 1$$

.....

..... (telescoping)

$$= a_{0,0} + i$$

$$= i.$$

Hence,

$$\lim_{i\to\infty} \frac{a_{i,0}}{i} = 1.$$

Claim : $\forall j \exists$ constant C_j such that

$$a_{i,j} = C_j + i(1-d)^j \quad \text{for } i > jd.$$

Proof: We prove our claim by induction on j.

Base case : For the case with $j = 0$, we choose $C_0 = 0$. Our claim for the base case is clearly satisfied.

Induction Hypothesis : Assume claim is true up to some j-1.

Inductive Step : By equation (1), we have for $j \geq 1$

$$a_{i,j} = a_{i-1,j} - a_{i-1,j-1} + a_{i-d,j-1}$$

$$= a_{i-1,j} - [C_{j-1} + (i-1)(1-d)^{j-1}]$$

$$+ [C_{j-1} + (i-d)(1-d)^{j-1}] \quad \text{for } i-d > (j-1)d$$

$$= a_{i-1,j} - (1-d)^{j-1}(i-1-i+d) \quad \text{for } i-d > (j-1)d$$

$$= a_{i-1,j} + (1-d)^{j} \quad \text{for } i > jd$$

$$\cdots\cdots$$

$$\cdots\cdots \quad \text{(telescoping } i - jd \textit{ times)}$$

$$= a_{jd,j} + (i-jd)(1-d)^{j}$$

$$= a_{jd,j} - jd(1-d)^{j} + i(1-d)^{j}.$$

Setting

$$C_j = a_{jd,j} - jd(1-d)^{j},$$

we have

$$a_{i,j} = C_j + i(1-d)^{j} \quad \text{for} i > jd. \qquad \text{Q.E.D.}\square$$

Moreover,

$$\lim_{i\to\infty} \frac{a_{i,j}}{i} = \lim_{i\to\infty} \frac{C_j + i(1-d)^{j}}{i}$$

$$= (1-d)^{j}.$$

Hence,

$$\lim_{i\to\infty} \frac{M_i}{i} = \lim_{i\to\infty} \frac{q \sum_{j=0}^{\infty} a_{i,j} q^{j}}{i}$$

$$= q \sum_{j=0}^{\infty} \lim_{i\to\infty} \frac{a_{i,j}}{i} \cdot q^{j}$$

$$= q \sum_{j=0}^{\infty} [q(1-d)]^{j}$$

$$= \frac{q}{1 - q(1-d)}$$

$$= \frac{q}{1 + (d-1)q} \,.$$

Therefore,

$$n = \frac{q}{1 + (d-1)q} \,.$$

Similarly,

$$n' = \frac{q_2}{1 + (d'-1)q_2} \,, \qquad \text{where } q_2 = 1 - p_2.$$

Proof of $\frac{\sqrt{Var[TT_A]}}{E[TT_A]} \rightarrow 0$ as $k \rightarrow \infty$

If we let the random variable X_i denote the number of burps that occur during the transmission of an i-packet long message, then we have the following recurrence:

$$X_i = ZX_{i-1} + \overline{Z}[1 + X_{i-d}] \quad \text{for } i \geq d,$$

where Z has the following distribution

$$Pr[Z = 1] = p,$$

$$Pr[Z = 0] = 1 - p,$$

and $\overline{Z} = 1 - Z$.

For $i \leq 0$, the distribution of X_i is

$$Pr[X_i = a] = \begin{cases} 1 & a = 0, \\ 0 & \text{otherwise.} \end{cases}$$

Let σ_i be the standard deviation of X_i, i.e., $\sigma_i = \sqrt{Var[X_i]}$. Also let σ_{TT_A} be the standard deviation of TT_A. We prove that $\lim_{k \to \infty} \frac{\sigma_{TT_A}}{E[TT_A]} = 0$ by first proving that $\lim_{i \to \infty} \frac{\sigma_i}{E[X_i]} = 0$. We begin with the following

Lemma.

Lemma 1. If A and B are two independent random variables, then,

$$Var[AB] = Var[A]Var[B] + E^2[B]Var[A] + E^2[A]Var[B].$$

Proof:

$$Var[AB]$$

$$= E[A^2B^2] - E^2[AB]$$

$$= E[A^2]E[B^2] - E^2[A]E^2[B] \quad \text{—(1).}$$

$$Var[A]Var[B]$$

$$= (E[A^2] - E^2[A])(E[B^2] - E^2[B])$$

$$= E[A^2]E[B^2] - E[A^2]E^2[B] - E^2[A]E[B^2] + E^2[A]E^2[B] \quad \text{—(2).}$$

(1) - (2) gives

$$Var[AB] - Var[A]Var[B]$$

$$= E[A^2]E^2[B] + E^2[A]E[B^2] - E^2[A]E^2[B] - E^2[A]E^2[B]$$

$$= (E[A^2] - E^2[A])E^2[B] + (E[B^2] - E^2[B])E^2[A]$$

$$= Var[A]E^2[B] + Var[B]E^2[A].$$

Lemma 1 follows. □

From the distribution of Z and $\overline{Z}$, we get

$$E[Z] = p, \qquad Var[Z] = p(1-p),$$

$$E[\overline{Z}] = 1 - p, \qquad Var[\overline{Z}] = p(1-p).$$

From the recurrence of X_i's, we have

$$X_i = ZX_{i-1} + \overline{Z}[1 + X_{i-d}] \quad \text{for } i \geq d,$$

and hence, for $i > d$,

$$Var[X_i] = Var[ZX_{i-1}] + Var[\overline{Z}(1 + X_{i-d})] + 2Cov[ZX_{i-1}, \overline{Z}(1 + X_{i-d})]\dagger$$

$$= Var[ZX_{i-1}] + Var[\overline{Z}(1 + X_{i-d})] +$$

$$2\{E[ZX_{i-1}\overline{Z}(1 + X_{i-d})] - E[ZX_{i-1}]E[\overline{Z}(1 + X_{i-d})]\}.$$

$$= Var[ZX_{i-1}] + Var[\overline{Z}(1 + X_{i-d})] - 2E[ZX_{i-1}]E[\overline{Z}(1 + X_{i-d})].$$

Using Lemma 1,

$$Var[X_i] = Var[Z]Var[X_{i-1}] + E^2[X_{i-1}]Var[Z] + E^2[Z]Var[X_{i-1}] +$$

$$Var[\overline{Z}]Var[X_{i-d}] + E^2[1 + X_{i-d}]Var[\overline{Z}] + E^2[\overline{Z}]Var[X_{i-d}] -$$

$$2E[Z]E[X_{i-1}]E[\overline{Z}]E[1 + X_{i-d}]\ddagger$$

$$= p(1-p)Var[X_{i-1}] + p(1-p)E^2[X_{i-1}] + p^2 Var[X_{i-1}] +$$

$$p(1-p)Var[X_{i-d}] + p(1-p)E^2[1 + X_{i-d}] + (1-p)^2 Var[X_{i-d}] -$$

$$2p(1-p)E[X_{i-1}]E[1 + X_{i-d}]$$

$$= pVar[X_{i-1}] + (1-p)Var[X_{i-d}] + p(1-p)(E[X_{i-1}] - E[1 + X_{i-d}])^2.$$

From recurrence

$$X_i = ZX_{i-1} + \overline{Z}[1 + X_{i-d}] \quad \text{for } i \geq d,$$

so,

$$E[X_i] = E[Z]E[X_{i-1}] + E[\overline{Z}]E[1 + X_{i-d}]$$

$$= pE[X_{i-1}] + (1-p)E[1 + X_{i-d}].$$

Since $E[X_i] \geq E[X_{i-1}]$,

---

† $Cov(X,Y)$ = covariance of X and Y = $E[XY] - E[X]E[Y]$.

‡ Z, X_{i-1} are independent, so as $\overline{Z}$ and X_{i-d}.

$$E[X_{i-1}] \leq pE[X_{i-1}] + (1-p)E[1 + X_{i-d}],$$

$$(1-p)E[X_{i-1}] \leq (1-p)E[1 + X_{i-d}],$$

$$(1-p)(E[X_{i-1}] - E[1 + X_{i-d}]) \leq 0,$$

$$E[X_{i-1}] - E[1 + X_{i-d}] \leq 0.$$

On the other hand,

$$\begin{aligned} E[X_{i-1}] - E[1 + X_{i-d}] &= E[X_{i-1}] - E[X_{i-d}] - 1 \\ &\geq -1 \qquad \text{(because } E[X_{i-1}] \geq E[X_{i-d}]\text{)}. \end{aligned}$$

Therefore,

$$(E[X_{i-1}] - E[1 + X_{i-d}])^2 \leq 1.$$

Hence,

$$Var[X_i] \leq pVar[X_{i-1}] + (1-p)Var[X_{i-d}] + p(1-p).$$

Let $v_i = \mathrm{Var}[X_i]$, we have,

$$\begin{aligned} v_i &\leq pv_{i-1} + (1-p)v_{i-d} + p(1-p) \\ &\leq p \max\{ v_{i-1}, v_{i-d} \} + (1-p)\max\{ v_{i-1}, v_{i-d} \} + p(1-p) \\ &\leq v_{i-\delta_1} + p(1-p), \qquad \text{—(3)} \end{aligned}$$

where δ_1 is either 1 or d depending on whether v_{i-1} or v_{i-d} is bigger.

Telescoping m times until the subscript of v on the right hand side of (3) gets down to some value h such that $1 \leq h \leq d$, we get,

$$v_i \leq v_h + mp(1-p).$$

Since each telescoping step decrements the subscript of v by at least 1, and since h is between 1 and d,

$$m \leq i - d.$$

Therefore,

$$v_i \le v_h + (i-d)p(1-p),$$

$$\sigma_i \le \sqrt{v_h + (i-d)p(1-p)},$$

$$\frac{\sigma_i}{E[X_i]} = \frac{\sigma_i/i}{E[X_i]/i}\,.$$

Now,

$$\lim_{i\to\infty} \frac{\sigma_i}{i} = 0 \quad \text{and,}$$

$$\lim_{i\to\infty} \frac{E[X_i]}{i} = n, \text{ a constant} > 0 \text{ (see last section of the Appendix).}$$

By L'hospital rule,

$$\lim_{i\to\infty} \frac{\sigma_i}{E[X_i]} = 0/n = 0.$$

Finally,

$$TT_A = kT_a/2 + X_{k/2}dT_a,$$

$$Var[TT_A] = d^2T_a^2Var[X_{k/2}],$$

$$\sigma_{TT_A} = dT_a\sigma_{k/2}, \quad \text{and}$$

$$E[TT_A] = kT_a/2 + dT_aE[X_{k/2}].$$

$$\frac{\sigma_{TT_A}}{E[TT_A]} = \frac{dT_a\sigma_{k/2}}{kT_a/2 + dT_aE[X_{k/2}]}$$

$$= \frac{d\,\dfrac{\sigma_{k/2}}{E[X_{k/2}]}}{\dfrac{k/2}{E[X_{k/2}]} + d}\,,$$

$$\lim_{k\to\infty} \frac{\sigma_{TT_A}}{E[TT_A]} = \frac{d\cdot 0}{\dfrac{1}{n} + d}$$

$= 0. \square$

References

1. U. D. Black, in *Data Networks : concepts, theory, and practice*, Prentice-Hall, 1989.
2. G.J. Coviello, ‘‘Comparative discussion of circuit vs. packet switched voice,’’ *IEEE Transaction of Communication*, vol. 27, pp. 1153-1160, 1979.
3. F. Cristian, ‘‘Probabilistic Clock Synchronization,’’ *Distributed Computing*, vol. 3:3, 1989.
4. D. Dolev and H.R. Strong, ‘‘Polynomial algorithms for multiple processor agreement,’’ *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pp. 401-407, 1982.
5. M.J. Fischer, N.A. Lynch, and M.S. Paterson, ‘‘Impossibility of distribute consensus with one faulty process,’’ *Journal of the ACM*, vol. 32, pp. 374-382, 1985.
6. H. Garcia-Molina, ‘‘Elections in a Distributed Computing System,’’ *IEEE Transaction on Computers*, vol. C(31), pp. 48-59, 1982.
7. H. Garcia-Molina, B. Kao, and D. Barbara, ‘‘Agressive Transmissions over Redundant Paths,’’ *Proceedings of the 11th International Conference on Distributed Computing Systems*, pp. 198-207, 1991.
8. J.N. Gray, in *Notes on Database Operating Systems. Operating Systems : An Advanced Course, Lectture Notes in Computer Science*, vol. 60, pp. 393-481, Springer-Verlag, 1978.
9. Horst, Robert, and T. Chou, ‘‘The Hardware Architecture and Linear Expansions of Tandem NonStop Systems,’’ *Tandem Technical Report 85.3*, April 1985.
10. K.Fitzgerald, ‘‘Vulnerability Exposed in AT&T’s 9-hour Glitch,’’ *IEEE The Institute* , vol. 14:3, March 1990.
11. L. Kleinrock, in *Queueing Systems*, vol. 2, John Wiley & Sons, 1976.
12. J.F. Kurose, M. Schwartz, and Y. Yemini, ‘‘Multiple-Access Protocols and Time-Constrained Communication,’’ *ACM Computing Surveys*, vol. 16, pp. 43-70, 1984.

13. L. Lamport, "Time, Clocks, and the Ordering of Events in a Distributed System," *Communication of the ACM*, vol. 21, pp. 558-565, 1978.

14. L. Lamport, "The mutual exclusion problem : part I — a theory of interprocess communication," *Communications of the ACM*, vol. 33, pp. 313-326, 1986.

15. L. Lamport, "The mutual exclusion problem : part II — statement and solutions," *Communications of the ACM*, vol. 33, pp. 327-348, 1986.

16. S. Lin and D. J. Costello, in *Error Control Coding : Fundamentals and Applications*, Prentice Hall, 1983.

17. D.A. Menasce and R.R. Muntz, "Locking and deadlock detection in distributed databases," *IEEE Transactions on Software Engineering*, vol. 5, pp. 195-202, 1979.

18. K.J. Perry and S. Toueg, "Distributed agreement in the presence of processor and communication faults," *IEEE Transaction of Software Engineering*, vol. 12, pp. 477-482, 1986.

19. P. Ramanathan and K. G. Shin, "A Multiple Copy Approach for Delivering Messages Under Deadline Constraints," *FTCS-21 (also in ACM TOCS)*, pp. 300-307, June, 1991 (May, 1992).

20. T.K. Srikanth and S. Toueg, "Optimal Clock Synchronization," *Journal of the ACM*, vol. 34, pp. 626-645, 1987.

21. J.A. Stankovic and K. Ramamritham, in *Hard Real-Time Systems*, 1988.

22. A.S. Tanenbaum, in *Computer Networks*, Prentice-Hall, 1988.

23. K.S. Trivedi, in *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*, Prentice-Hall, 1982.