Chapter 2

Suppression

In this chapter, we present an analysis of the basic Suppression (SUP) algorithm. We describe the behavior of the algorithm using pseudocode. We discuss why it is considered a scalable technique for groups of communicating processes, and under what conditions it is most useful to employ.

We introduce two key performance metrics to describe the Suppression algorithm: Time Elapsed, the delay incurred by a process that employs the Suppression technique; and Extra Messages, the corresponding messaging overhead. We present the intuition behind the derivation of these metrics, then explore the impact of different Suppression timer distribution functions on the metrics. We examine a uniform distribution, as well as a decaying exponential distribution. We also investigate the creation of designer distribution functions that optimize a particular metric, in those instances where a subset of the network parameters remain fixed.

By studying deployed protocols and applications, we are able to use realistic values for various network parameters that affect the metrics. Consequently, we estimate operating ranges for the metrics and use these ranges as input to simulations. We present simulation results that validate the analysis. In addition, we provide an overview of related work, a summary of our findings, and proposals for future avenues of research.

In Chapter 3, we revisit the analysis of Suppression under lossy network conditions. In Chapter 5, we explore the use of Suppression as a building block for other algorithms; we examine its role in the composition of the Leader Election algorithm.

2.1 Core Algorithm

The general behavior of the algorithm is described in this section using pseudocode (Program 2.1). There are N processes[1] numbered $0, \ldots, N-1$. When a process i arrives in the system, it chooses a random time t to delay sending its message by calling `random`(d, T). Process i selects time t

[1] In general, we assume the processes reside on N distinct processors across the network. However, Suppression is still useful even when the processes are co-resident on a single processor. If processes awaken simultaneously, the processor still may wish to spread out and to aggregate responses in order to handle the load.

with random probability as described by the density d. If the distribution is uniform, t is randomly selected from the uniform interval $[0, T]$, whereas if it is exponential, t is selected from a decaying exponential distribution with decay parameter $\alpha = T$ (Section 2.4.2). The process then calls the `sleep`(t) routine, causing the process to wait for t units of time to elapse. On awakening, the process checks if it has received a message from any other processes, $j \neq i$, during the delay interval. If there were `no_messages_received`(), process i sends its message. Otherwise, the message is suppressed.

```
SUPPRESSION(d, T)
1   t = random(d, T)
2   sleep(t)
3   if no_messages_received() then
4     send_message()
```

Program 2.1: **Suppression Algorithm.**

2.2 Scalability

The Suppression (SUP) algorithm is considered scalable because it decreases the number of messages generated by a group of communicating processes. SUP takes what would have been simultaneous message transmissions and spreads them out over the suppression interval, making message implosion less likely. At the same time, early messages suppress the messages that would have been sent by later awakening processes. Therefore, SUP reduces the overall number of messages sent, ideally replacing multiple message transmissions with a single message. The end result is that Suppression helps avoid network congestion, processor overrun, and packet loss.

Suppression is an important tool to employ when all processes generate similar messages, which otherwise would lead to redundancy. Messages generated by different processes can be replaced by a single message because the content of the messages is identical.

Suppression may be needed when, for example, process actions are synchronized in time: when processes arrive for a scheduled event, such as a teleconference; when all processes respond to the same multicast message received, such as a query sent to a cluster of servers; and when all processes bootstrap themselves at initialization or after a failure.

2.3 Metrics

We can analyze the effect of Suppression in the worst case when all N processes *arrive* into the system simultaneously. Simultaneous arrivals are considered a worst case scenario because that is when there is the greatest potential for concurrent message transmission. When the number of

processes grows large, there are potentially even greater bottlenecks due to synchronized competition for resources, such as network bandwidth, CPU cycles (to handle message interrupts), or memory for the storage of process state.

The scalability afforded by Suppression is not without a cost. What is gained by reducing messaging overhead is at the expense of timeliness. Therefore, the effectiveness of the Suppression algorithm is measured using two metrics: the time before the first message is sent and the number of extra messages generated. The former gauges the latency of the algorithm, whereas the latter gauges the message overhead. The challenge is to minimize the delay but also to minimize the number of messages transmitted.[2]

We explore these metrics in the sections below. Our approach is to define the metrics analytically using probability theory, then to examine the effects of allowing the random delay variable to be selected using different distributions. We use simulation to validate our results since approximations are used to make the analysis tractable.

For simplicity, the metrics are initially studied in the context of a network without message loss, where the transmission delay between processes, Δ, is fixed. In Chapter 3, the model is extended to consider loss.

2.3.1 Time Elapsed

The Time Elapsed metric measures the minimum time t_i selected by any process. The minimum time selected also can be construed as the minimum delay before all other processes have the potential to become suppressed. In some sense, this can be viewed as the minimum potential *stopping time* of the algorithm, or its earliest opportunity to complete. Completion is important, as Suppression is often used as an initial mechanism to reduce the pool of participating processes (i.e., processes issuing messages), after which another algorithm is performed among the remaining processes (IGMP [24], PIM [23], SRM [27]).

Expected Value. The expected value of a random variable x is $E[x] = \sum xp(x)$, where $p(x)$ is the probability density function. The expected value of t_{min}, the minimum elapsed time before one of the processes wakes up and sends a message, is derived as follows.

Process i picks time t, using a probability density function $p_i(t)$, where $0 \leq t \leq T$ and T is the maximum amount of time a process waits before issuing a message. $P_i(t)$ is the cumulative distribution function, i.e., $P_i(t) = \int_0^t p_i(x)\,dx$.

The time chosen by process i is the minimum if every other process picks a time greater than t. Thus, in the second step of the derivation below, the sum contains N terms, and the product

[2]The time until the last message is sent is also an interesting metric, but more so when loss is introduced. We discuss it in detail in Chapter 3. In the lossless case, the upper bound on the time until the last message is simply the delay until the first message is sent plus the maximum transmission delay between any pair of participating process.

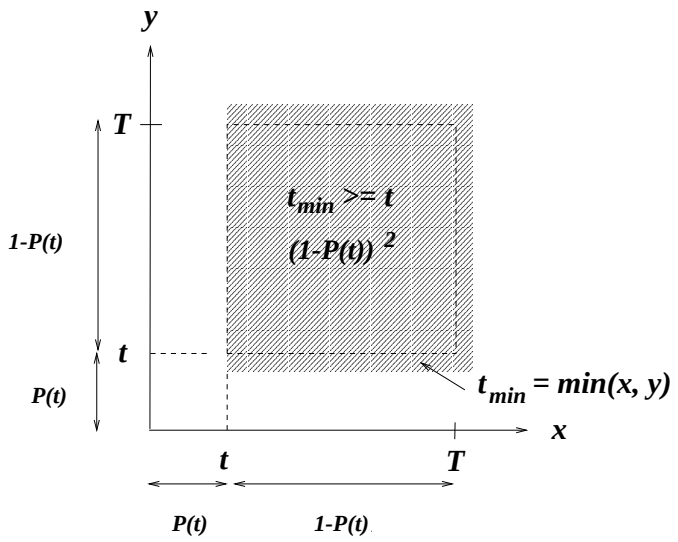

Figure 2.1: **Intuition Behind $E[t_{min}]$ for 2 Processes.**

contains $N-1$ terms. We add up the probabilities for each i, which gives us the likelihood of the minimum being equal to t. We assume that all density functions are the same, which allows us to drop the subscripts i and j between the second and third steps of the derivation.

$$\begin{aligned} E[t_{min}] &= E\Big[\min_{0\le i<N} t_i\Big] \\ &= \int_0^T t \sum_{0\le i<N} p_i(t) \prod_{j\neq i} \int_t^T p_j(x)\,dx\,dt \\ &= \int_0^T (1-P(t))^N\,dt \end{aligned}$$

Intuition. The intuition behind this formula is displayed in two-dimensions in Figure 2.1. Consider two processes, shown as two independent random variables x and y, where the cumulative distribution function is $P(t) = t/T$. Each process selects a time to delay. The minimum value is $t_{min} = min(x,y)$. The probability that $x \le t$ is $P(t)$. Therefore, the probability that $x \ge t$ is $1-P(t)$. Since x and y are independent variables, the probability that both values x and y are greater than or equal to t is the same as the probability that $t_{min} \ge t$. The result is equal to $(1-P(t))^2$, which is shown by the shaded region.

This construction generalizes to N dimensions. When there are N processes participating in the algorithm, the result is the area defined by $(1-P(t))^N$, which is exactly the integral proposed earlier. Although Figure 2.1 depicts a uniform distribution, the results generalize to other distributions as well.

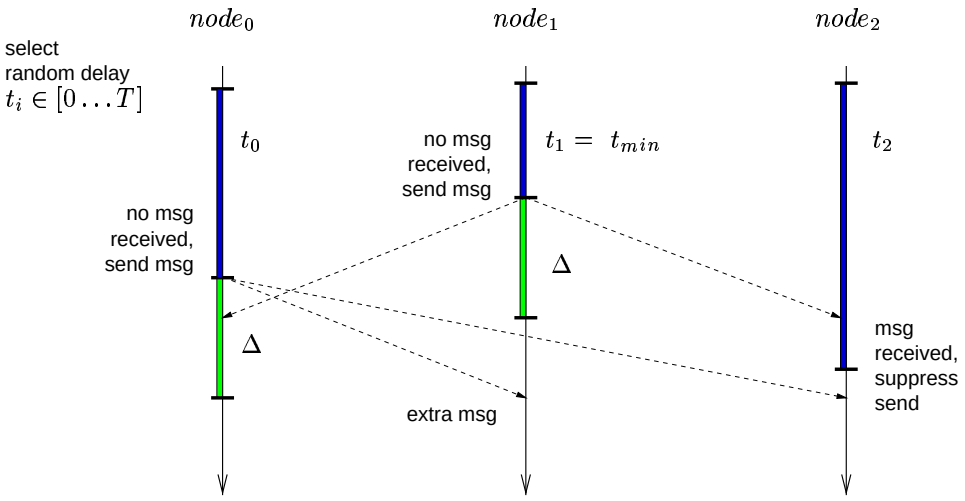

Figure 2.2: **Suppression Algorithm.**

2.3.2 Extra Messages

In a lossless network without transmission delay, only one message is generated by the Suppression algorithm. The earliest message generated suppresses all other messages.

In a lossless network with delay, any message generated beyond the earliest message is considered an extra message. Extra messages are generated because messages take a non-zero time to reach listeners. The earliest message is sent at time t_{min}, the minimum t_i, $0 \leq i < N$. A process receives this message at time $t_{min} + \Delta$, where Δ is the message transmission delay. Any message sent at time t_k such that $t_{min} < t_k < t_{min} + \Delta$ corresponds to an extra message that was not suppressed.

In Figure 2.2, three processes arrive into the system at the same time. Each selects a time t_i to delay. Transmission delay between all pairs of nodes equals Δ. The minimum delay t_{min} is chosen by $node_1$. Therefore, when $node_1$ does not receive a message during the interval $[0, t_1]$, it awakens and issues a message. When $node_2$ awakens after t_2 units of time, the process discovers it received a message from $node_1$, therefore suppressing its own message. However, on awakening at time t_0, $node_0$ has not yet received the message sent by $node_1$ because $t_{min} < t_0 < t_{min} + \Delta$. Therefore, $node_0$ sends its message, which is considered an extra message. Note that the number of extra messages is a count of the extra messages sent (not received).

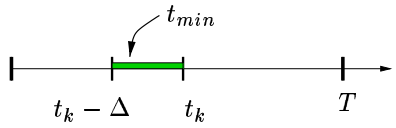

Figure 2.3: **Condition for Extra Message Transmission.**

Expected Value. The expected number of extra messages sent is calculated as follows. The expected number of extra messages is simply the sum of the likelihood of each process sending an extra message, or the number of processes that have selected a delay time that is within Δ of the minimum time selected. In other words, a process for which $t_k - \Delta < t_{min} < t_k$ holds true is a process that generates an extra message (Figure 2.3). The likelihood that the minimum lies between $t_k - \Delta$ and t_k is calculated by determining the likelihood that each process $i \neq k$ picks a time greater than $t_k - \Delta$, and subtracting from it the likelihood that every process $i \neq k$ picks a time greater than t_k.

$$\begin{aligned} E[\#\ extra] &= \sum_{0 \leq k < N} Pr\left[t_k - \Delta < t_{min} < t_k\right] \\ &= N \cdot P(\Delta) - 1 + N \int_{\Delta}^{T} p(t)(1 - P(t - \Delta))^{N-1}\, dt \end{aligned}$$

Intuition. The intuition behind this calculation is shown in Figure 2.4. Consider the problem with 2 processes, represented as two independent random variables x and y, where the cumulative distribution function is $P(t) = t/T$. The probability that process x chooses a suppression value $x \geq t$ is $1 - P(t)$, and the probability that process x chooses a suppression value $x \geq (t - \Delta)$ is $1 - P(t - \Delta)$. Therefore, the probability that x lies between $t - \Delta$ and t is equivalent to $(1 - P(t - \Delta)) - (1 - P(t)) = P(t) - P(t - \Delta)$.

The probability that $t_{min} = min(x, y)$ lies between t and $t - \Delta$ requires we perform this calculation in both dimensions, shown by the shaded area in the diagram. This result generalizes to N dimensions.

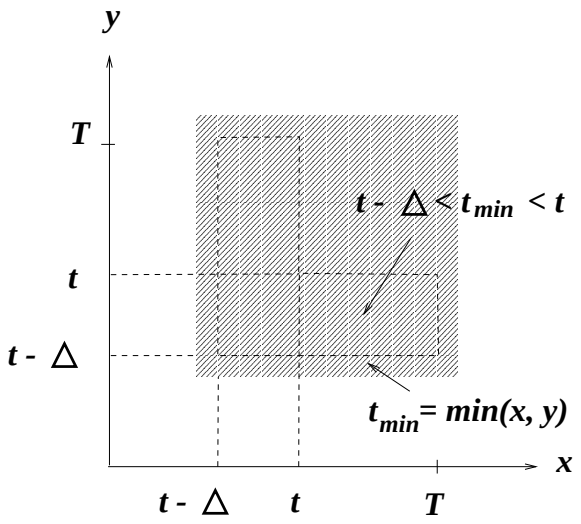

Figure 2.4: **Intuition Behind $E[\#extra]$ for 2 Processes.**

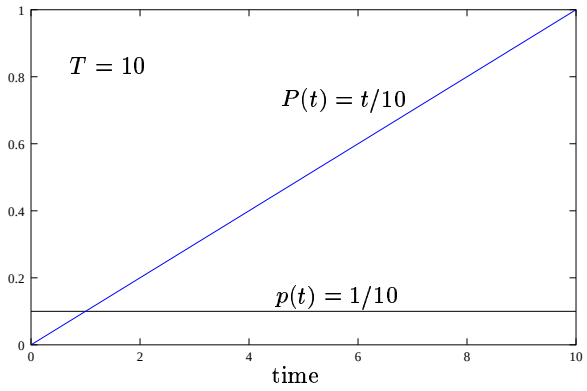

Figure 2.5: **Uniform Distribution.**

2.4 Distributions

In this section, we apply the analysis derived above to the uniform distribution and the decaying exponential distribution. The uniform distribution is attractive from the perspective that it is used in most deployed Suppression algorithms. It is also simple to implement and more efficient than implementations of exponential distributions. The exponential distribution is interesting from the standpoint that it has been used as an integral part of collision-avoidance algorithms for shared media, like Ethernet and packet radio. In addition, it is used as the basis for collision back-off schemes for shared media networks, among others.

2.4.1 Uniform Distribution

The most common probabilistic model in existing suppression algorithms is the uniform distribution. Each process randomly selects a value t from a uniform distribution in the range $[0 - T]$. Note that there is equal likelihood that any process will pick t_{min}; all processes are treated equally.

In Figure 2.5, we show a uniform distribution, with $p(t) = 1/T$, and $P(t) = t/T$. When the upper bound of the suppression interval is set to $T = 10$, the probability density of a process selecting a specific wake-up time t is $p(t) = 1/10$ and the cumulative probability for selecting t is $P(t) = t/10$.

$$\begin{aligned} E[t_{min}] &= \int_0^T (1 - t/T)^N \, dt \\ &= \left\{ \frac{-T(1 - t/T)^{N+1}}{N+1} \right|_0^T \\ &= \frac{T}{N+1} \end{aligned}$$

Consider one process. Intuitively, $E[t_{min}] = \frac{T}{2}$ makes sense. Averaging over many runs, the minimal suppression timer converges to the midway point, $T/(N+1)$, of the suppression interval, T. For a

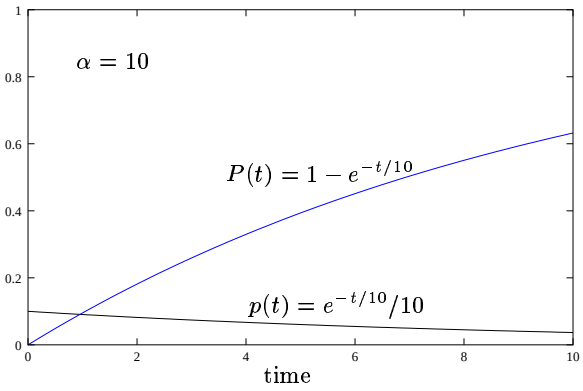

Figure 2.6: **Exponential Distribution.**

given N, it is more likely that one of the timer values selected will be less than the minimum timer value for the $N-1$ case, which is why the minimum timer value, on average, dissects the timer interval by $(N+1)$ since there exists one more random variable, which can be greater or less than t_{min}. If the new value is greater than t_{min}, the minimal suppression value is unchanged. However, if the new value is less than t_{min}, then t_{min} would decrease. Because there is a non-zero probability of the new value being less, t_{min} decreases as N increases.

$$\begin{aligned} E[\#\ extra] &= N\frac{\Delta}{T} - 1 + N\int_{\Delta}^{T} 1/T(1 - t/T + \Delta/T)^{N-1}\,dt \\ &= N\frac{\Delta}{T} - \left(\frac{\Delta}{T}\right)^{N} \end{aligned}$$

Consider what we can deduce about the behavior of the metric for extra messages. When $N = 2$, the algorithm generates Δ/T extra messages on average. The intuition behind this result is that when there are two processes, one wakes up earliest at t_{min} and the other wakes up within Δ of t_{min} Δ/T of the time.

2.4.2 Decaying Exponential Distribution

For the case of the decaying exponential distribution, $p(t) = e^{-t/\alpha}/\alpha$, $P(t) = 1 - e^{-t/\alpha}$. In Figure 2.6, we set $\alpha = 10$, resulting in $p(t) = e^{-t/10}/10$, $P(t) = 1 - e^{-t/10}$.

$$\begin{aligned} E[t_{min}] &= \int_{0}^{\infty} (e^{-t/\alpha})^{N}\,dt \\ &= \frac{\alpha}{N} \end{aligned}$$

$$
\begin{aligned}
E[\#\ extra] &= N(1-e^{-\Delta/\alpha}) - 1 + N\int_{\Delta}^{\infty} \frac{e^{-t/\alpha}}{\alpha}(e^{-(t-\Delta)/\alpha})^{N-1}\,dt \\
&= (N-1)(1-e^{-\Delta/\alpha})
\end{aligned}
$$

2.5 Analysis

In this section, we explore realistic values for T, N, and Δ, as well as the ratios T/N and Δ/T, which appear prominently in the derivation of the metrics. The parameter ranges are gleaned from the observation of deployed protocols and applications. Setting realistic parameter bounds serves two purposes; we can examine the operating range of the metrics and use these ranges as input to our simulations. We also study the impact of a distribution choice on the metrics. We analyze the uniform and exponential distributions separately, then compare the operational crossover points for each metric. Finally, we present our simulation results, which fully validate the analysis.

2.5.1 Realistic Parameters

For existing algorithms, the suppression interval, T, takes on a value of $O(1)$ second for local updates and quickly changing data (SRM [27]); $O(5)$ seconds for real-time control status updates (RTCP [54]); $O(10)$ seconds for multicast group management (IGMP [24]); $O(.5)$ - $O(1)$ minutes for router and QoS protocols (PIM [22] [17] and RSVP [61] [60] [8]); $O(5)$ minutes for slowly changing data updates and updates using adaptive periodicity to maintain fixed bandwidth. Some of these values represent announcement periodicities rather than suppression intervals. They are included here because they measure the relative frequency of process interactions in system-wide multicast algorithms. We display the timer ranges in Table 2.1.

N, the number of processors participating in the algorithm, ranges from $O(1)$ for DNS server load balancing, distributed file systems, and personal conferencing, to $O(10)$ for ISP mail servers and Web gateways, to $O(100)$ for parallel computations, to $O(1000)$ for Internet telepresentations [3], and $O(10K)$ for the number of processors that might simultaneously request a Web page from a popular site (Table 2.2).

An estimate of Δ, the transmission delay, depends on the domain of communication. The range is quite large when we compare Ethernet performance to that of the wide-area Internet. The variance is also pronounced when comparing the local Ethernet delay with that of satellite or modem communication, both considered slow links. Our estimated values for delay are .1-.5 msec for the local subnet (Ethernet), $O(1)$ msec local but different subnets (e.g., 2-3 msec across a bridge), $O(10)$ msec to transit a router or series of routers (note that a delay-free transmission takes approximately 30 msec to transit the continental US, based on the speed of light), $O(100)$ msec for slow connections within the US and communications between continents and beyond, $O(500)$ msec

for roundtrip Satellite communication, and finally $O(10)$ seconds and upwards for Interplanetary Internet transmissions (Table 2.3). The times reported include the time for a message to transit the network, as well as to get through the operating system network protocol stack.

$O(T)$	*Protocol*
1 sec	SRM
5 sec	RTP
10 sec	IGMP
30 sec	RSVP
60 sec	PIM
5 min	DNS, MZAP

Table 2.1: **Timer Range (T).**

$O(N)$	*System*
10^0	Distributed file systems, Personal conferencing
10^1	ISP mail servers, Web server load balancing
10^2	Parallel computations
10^3	Internet telepresentations
10^5	Simultaneous Web page requests

Table 2.2: **Number of Processes (N).**

$O(\Delta)$	*Network*
.1 msec	Same Subnet
1 msec	Local but Different Subnet
10 msec	Across a Router
30 msec	Speed of Light across US
100 msec	Moderate US connection
500 msec	Satellite transmission
10 sec	Interplanetary Internet

Table 2.3: **Transmission Delay (Δ).**

We assume $T \geq 1$, $N \geq 1$, and $\Delta \leq T$. Therefore $\Delta/T \leq 1$ and has a positive range of $[3e\text{-}06, .5]$, whereas T/N has a positive range $[1e\text{-}05, 300]$.

2.5.2 Uniform Distribution

When using a uniform distribution for the Suppression timer, the expected delay before which a message is sent, $E[t_{min}]$, is simply a function of T and N. Using a log scale on the y-axis, we plot $E[t_{min}]$ vs. T in Figure 2.7, and using a log-log plot, we show $E[t_{min}]$ vs. N in Figure 2.8. Simply

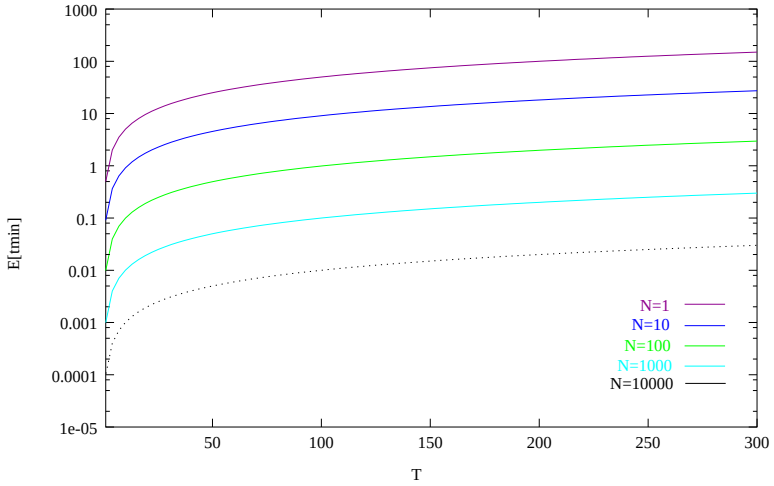

Figure 2.7: **Uniform: Time Elapsed vs. T.**

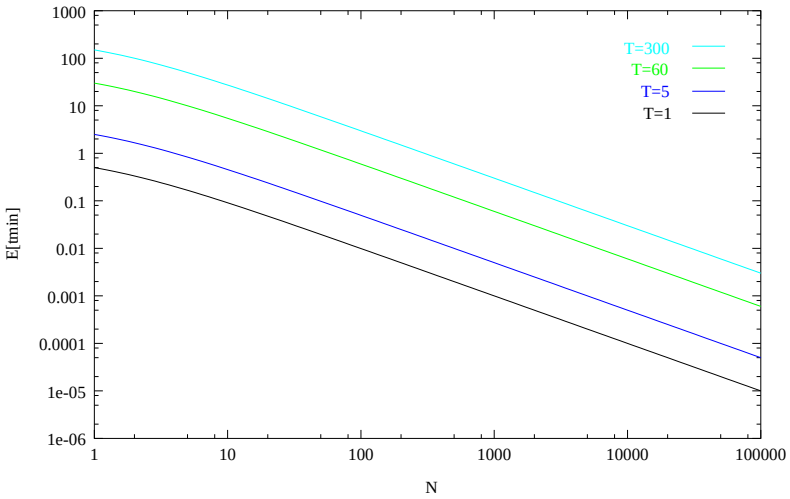

Figure 2.8: **Uniform: Time Elapsed vs. N.**

put, $E[t_{min}]$ grows with T, the upper bound on the suppression range, and shrinks with N, the number of processes participating in the algorithm.

The relationship between the elapsed time $E[t_{min}]$ and the variables T and N is even more clearly evident as N grows very large,

$$\begin{aligned} E[t_{min}] &= \frac{T}{N+1} \\ &\approx \frac{T}{N} \end{aligned}$$

As N grows very large, $E[t_{min}]$ approaches 0.

The ratio Δ/T plays a critical role in the expected number of extra messages sent, $E[\#\ extra]$.

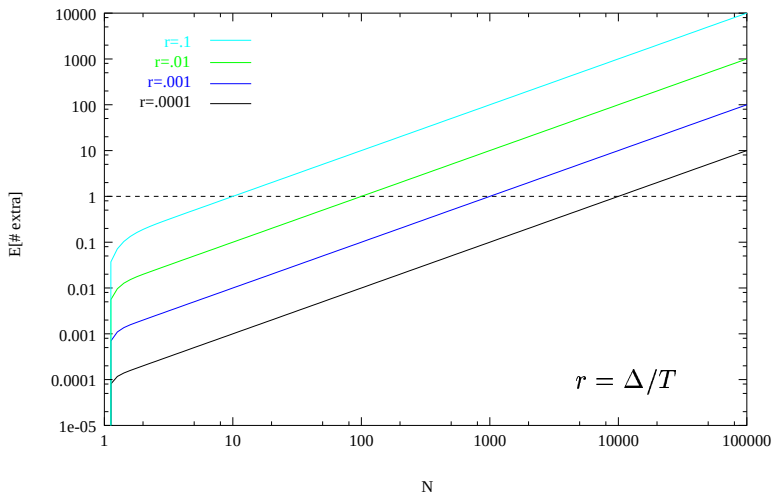

Figure 2.9: **Uniform: Extra Messages vs. N.**

Using a log-log scale, Figure 2.9 plots $E[\#\ extra]$ vs. N and displays the effects of the ratio $r = \Delta/T$ over several orders of magnitude. Figures 2.10 and 2.11 display $E[\#\ extra]$ vs. Δ/T with large and small N respectively.

As N grows very large,

$$\begin{aligned} E[\#\ extra] &= N\left(\frac{\Delta}{T}\right) - \left(\frac{\Delta}{T}\right)^N \\ &\approx N\left(\frac{\Delta}{T}\right) \end{aligned}$$

Statistically it will take $N \geq T/\Delta$ processes to generate extra messages. Therefore, when N is very large, Δ/T must be extremely small to avoid extra messages. When N is small, the ratio between Δ and T can afford to be larger. Thus the relationship between N and Δ/T is important for proper system parameterization.

This observation suggests that suppression may not be effective at keeping the number of extra messages low for extremely large groups, especially in contexts like the wide-area Internet where it may be hard or even impossible to keep the Δ/T ratio predictably low, and hence to keep large numbers of extra messages from being generated. Extra messages are also a concern for bandwidth-limited networks (e.g., modem channels, wireless sensor nets), which cannot afford the overhead.

Note that when the metrics $E[t_{min}]$ and $E[\#\ extra]$ are examined with N set to a large value, their inverse relationship becomes clear. When the parameters that influence the expected number of extra messages are re-arranged,

$$E[\#\ extra] \approx N(\Delta/T) = \Delta(N/T) \approx \Delta/E[t_{min}]$$

There is a direct tradeoff between reducing the number of extra messages generated and increasing the minimum delay before which the first message is sent. The metrics are inversely proportional by a factor of Δ.

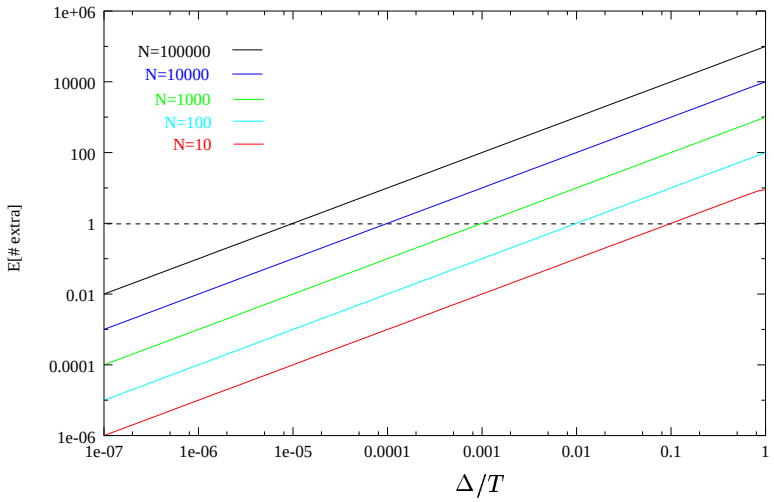

Figure 2.10: **Uniform: Extra Messages vs. Δ/T (Large N).**

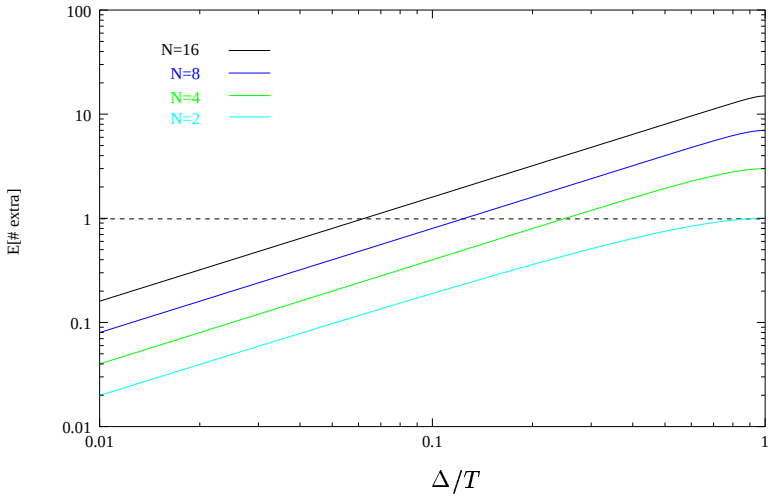

Figure 2.11: **Uniform: Extra Messages vs. Δ/T (Small N).**

While a given system parameterization might improve one metric, it is **always** at the expense of the other. Therefore, the optimal balance will be a function of the operating environment. For example, if the context is a group of processes communicating over a network with limited capacity or a group of sensors with limited power for communication, the goal might be to keep $E[\#\ extra]$

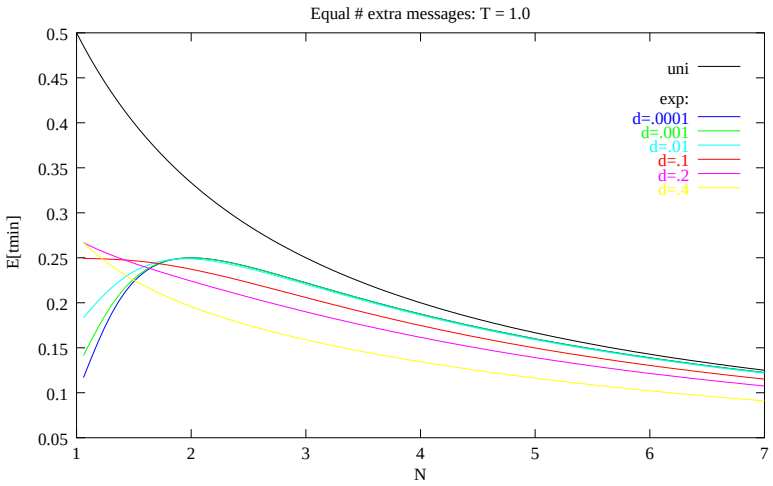

Figure 2.12: **Comparison: Time Elapsed vs. N (Varying $d = \Delta$).**

below a given threshold. If N or Δ fluctuates, then T will need to be adaptive to compensate. We elaborate on this point in Section 2.7.

2.5.3 Decaying Exponential Distribution

When using an exponential distribution for the Suppression timer, the expected delay before which a message is sent, $E[t_{min}]$, is now a function of α and N. As N grows very large, $E[t_{min}]$ behaves like the Uniform distribution, with $\alpha = T$.

$$E[t_{min}] \approx \frac{\alpha}{N}$$

Similarly, the ratio Δ/α plays a critical role in $E[\#\ extra]$. As N grows large,

$$\begin{aligned} E[\#\ extra] &= (N-1)(1-e^{-\Delta/\alpha}) \\ &\approx N\left(1-e^{-\Delta/\alpha}\right) \end{aligned}$$

2.5.4 Comparisons

Below, the performance of the uniform and decaying exponential distributions are compared. The cross-over points for $E[t_{min}]$ and $E[\#\ extra]$ are determined by setting the metrics for the two distributions equal under certain conditions.

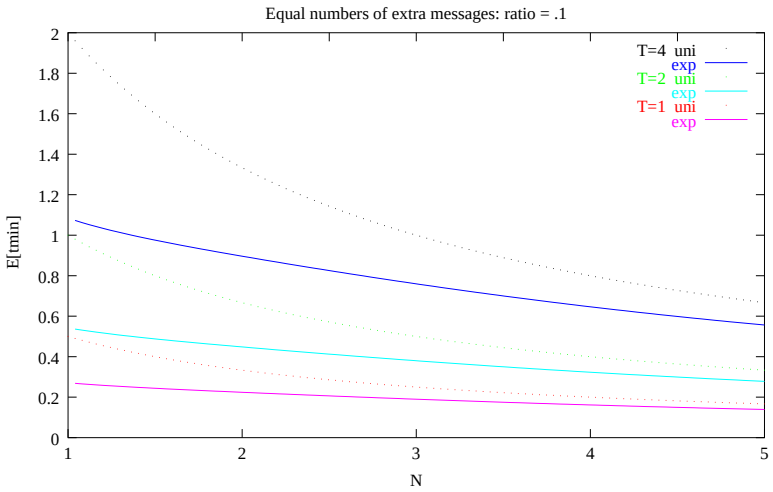

Figure 2.13: **Comparison: Time Elapsed vs. N (Varying T).**

Time Elapsed. The decaying exponential distribution generates the same average number of extra messages as the uniform distribution if we set $E[\#\ extra]$ equal for the two distributions, solve for α, then plug α back into the original equation to rederive $E[t_{min}]$ for the decaying exponential distribution,

$$\begin{aligned} N\left(\frac{\Delta}{T}\right) - \left(\frac{\Delta}{T}\right)^N &= (N-1)(1-e^{-\Delta/\alpha}) \\ \alpha &= -\Delta/\ln\left(1 - \frac{N(\Delta/T) - (\Delta/T)^N}{N-1}\right) \\ E[t_{min}] &= \frac{\alpha}{N} \\ &= -\Delta/N \ln\left(1 - \frac{N(\Delta/T) - (\Delta/T)^N}{N-1}\right) \end{aligned}$$

When $E[t_{min}]$ vs. N is plotted while keeping the number of extra messages equal, the exponential distribution is marginally more responsive than the uniform distribution. This phenomenon is more pronounced for small N and larger ratios, $.1 < \Delta/T \leq 1$ (Figure 2.12), as well as for larger T (Figure 2.13).

Extra Messages. The decaying exponential distribution generates the same minimum delay as the uniform distribution if we set $E[t_{min}]$ equal for the two distributions. To rederive $E[\#\ extra]$ for the decaying exponential distribution, we solve for α, then plug it back into the original equation. The last step is reached by using the Taylor expansion $e^x = 1 + x + x^2/2! + x^3/3! + \ldots \geq 1 + x$, and noting that $1 - e^x \leq -x$.

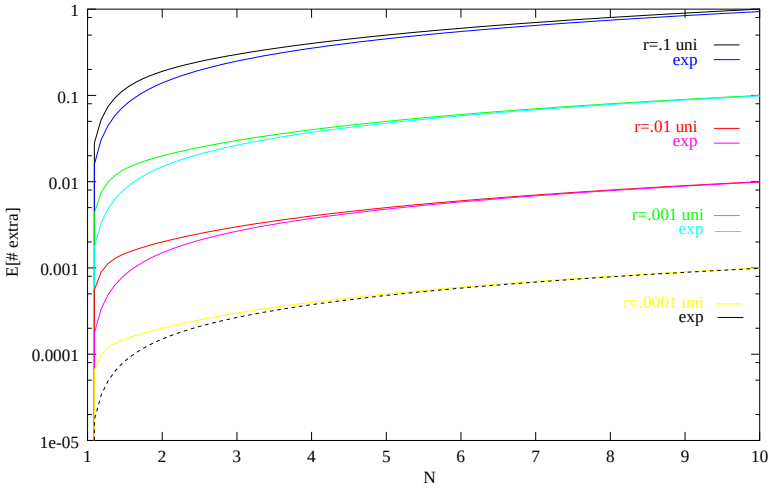

Figure 2.14: **Comparison: Extra Messages vs. N (Small $r = \Delta/T$).**

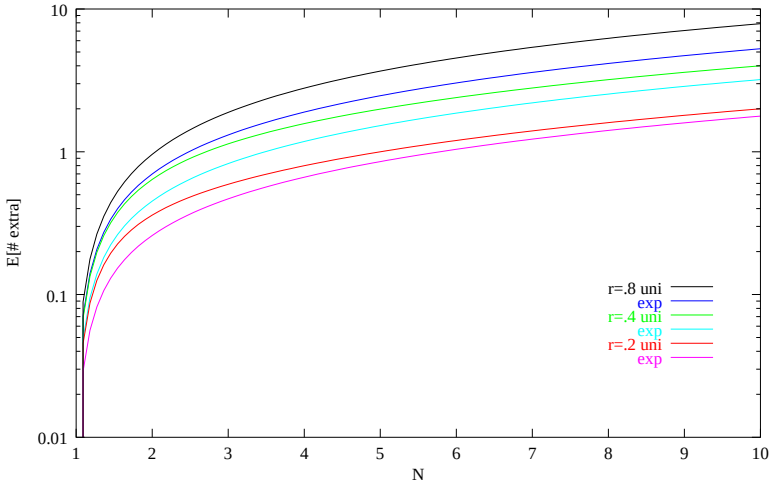

Figure 2.15: **Comparison: Extra Messages vs. N (Large $r = \Delta/T$).**

$$
\begin{aligned}
\frac{T}{N+1} &= \frac{\alpha}{N} \\
\alpha &= TN/(N+1) \\
E[\#\ extra] &= (N-1)(1-e^{-\Delta(N+1)/TN}) \\
&\leq \frac{(N-1)\Delta(N+1)}{TN} = N\left(\frac{\Delta}{T}\right) - \frac{1}{N}\left(\frac{\Delta}{T}\right)
\end{aligned}
$$

Under these conditions, the exponential distribution always outperforms the uniform distribution in terms of E[# messages]. However, with small Δ/T, the impact is fewer than one extra message, and the exponential and uniform curves quickly converge beyond small N. With larger ratios, the effects are significant, especially as N increases. These results are evident in Figures 2.14 and

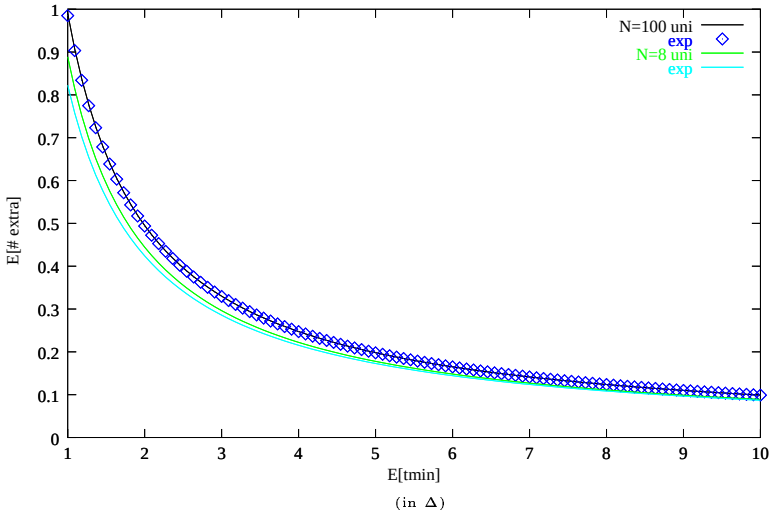

Figure 2.16: **Comparison: Extra Messages vs. Time Elapsed (Large N).**

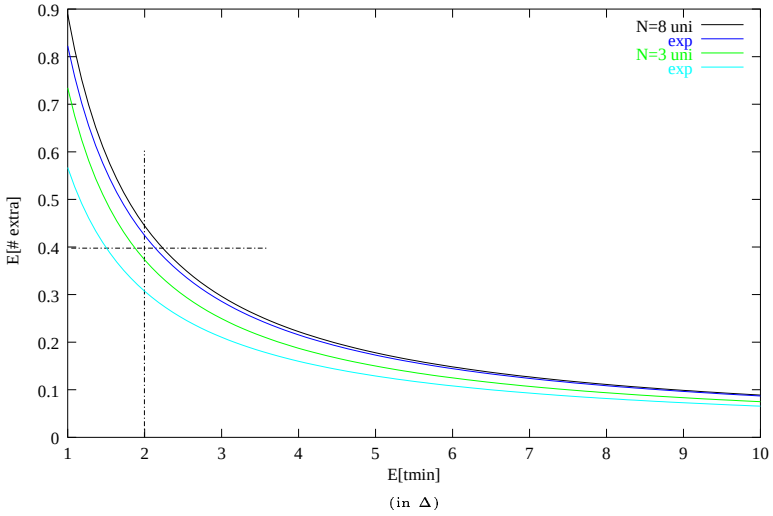

Figure 2.17: **Comparison: Extra Messages vs. Time Elapsed (Small N).**

Figure 2.15, which display $E[\#\ extra]$ vs. N for uniform and exponential distributions with small and large ratios of Δ/T respectively. The results raise the issue that it may make sense to design adaptive systems that use one distribution in one part of the operating range and another when it crosses into another operating range. For example, when $\Delta/T \leq .1$ use a uniform distribution, whereas if $\Delta/T > .1$ use a decaying exponential distribution. Current multicast protocols employing the Suppression technique strictly rely on uniform distributions, meaning that if an application has a requirement to minimize messaging overhead (as compared to responsiveness), then it is not having its needs met adequately. In addition, this raises the larger issue of whether or not to explore other distributions.

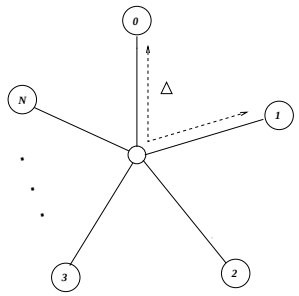

Figure 2.18: **Star Topology with fixed Δ.**

Extra Messages vs. Time Elapsed. In Figure 2.16 and Figure 2.17, $E[\#\ extra]$ vs. $E[t_{min}]$ is plotted for uniform and exponential distributions, with large and small N, respectively. Units of Δ are derived for the uniform distribution by observing that $E[t_{min}] = T/N + 1$. Therefore, $T = (N+1)E[t_{min}]$. Substituting for T in $E[\#\ extra]$ and setting $x = E[t_{min}]/\Delta$, we derive:

$$\begin{aligned} E[\#\ extra] &= N\left(\frac{\Delta}{(N+1)E[t_{min}]}\right) - \left(\frac{\Delta}{(N+1)E[t_{min}]}\right)^N \\ &= N\left(\frac{1}{(N+1)x}\right) - \left(\frac{1}{(N+1)x}\right)^N \end{aligned}$$

The graphs display the tradeoff between $E[\#\ extra]$ and $E[t_{min}]$ in units of Δ for different N. The probability of producing extra messages diminishes as delay increases. Regardless of Δ, one observes which curves have better responsiveness, $E[t_{min}]$, by drawing a horizontal line at a constant $E[\#\ extra]$. Similarly, to obtain which curves have less overhead, $E[\#\ extra]$, one draws a vertical line at a constant $E[t_{min}]$. Note that both graphs reveal that for smaller N, the decaying exponential distribution outperforms the uniform distribution. By $N = 8$, the differences are small, and by $N = 100$, imperceptible.

2.6 Simulation

In our initial simulations, N, T and Δ were fixed. The network topology was configured so that the transmission delay between all pairs of nodes was identical. This was accomplished by creating a star topology (Figure 2.18), with the processes at the edges of the star, and with Δ calculated as specified below.

$$\Delta = \frac{s}{b} + d$$

$$\begin{aligned} s &= \text{packet size in bits} \\ b &= \text{link speed in bits per second} \\ d &= \text{link delay in seconds} \end{aligned}$$

Each simulation was run 1000 times using the ns simulator [6]. The range for N was examined from 1 to 10 in increments of 1, and from 10 to 100 in increments of 10. For a few specific data sets, we also examined N between 100 and 1000 in increments of 100.

Each node chose a delay time, t_i, based on the Unix `srandom()` function that had been seeded with the simulation start time. All simulation results successfully overlay the analysis results. As can be seen, the simulation for $E[t_{min}]$ vs. T matches (Figure 2.19) as does the simulation comparing $E[t_{min}]$ vs. N (Figure 2.20). Similarly, the simulations of $E[\#extra]$ vs. N and $E[\#extra]$ vs. Δ/T validate the analysis (Figures 2.21, 2.22, 2.23).

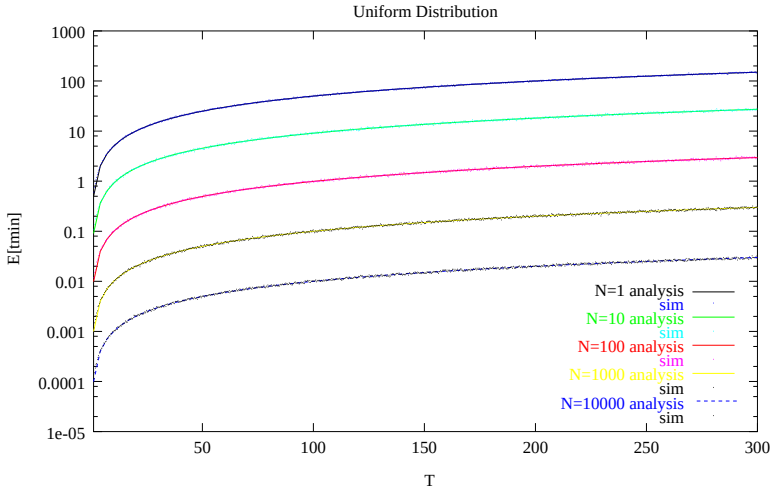

Figure 2.19: **Simulation vs. Analysis: Time Elapsed vs. T (Uniform).**

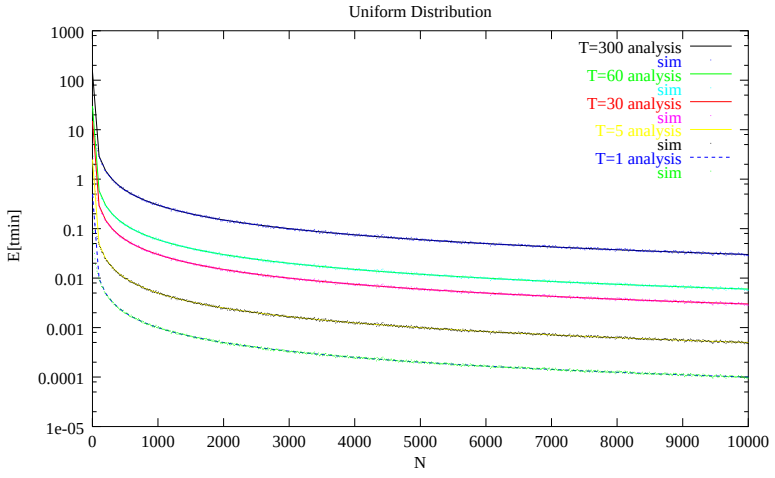

Figure 2.20: **Simulation vs. Analysis: Time Elapsed vs. N (Uniform).**

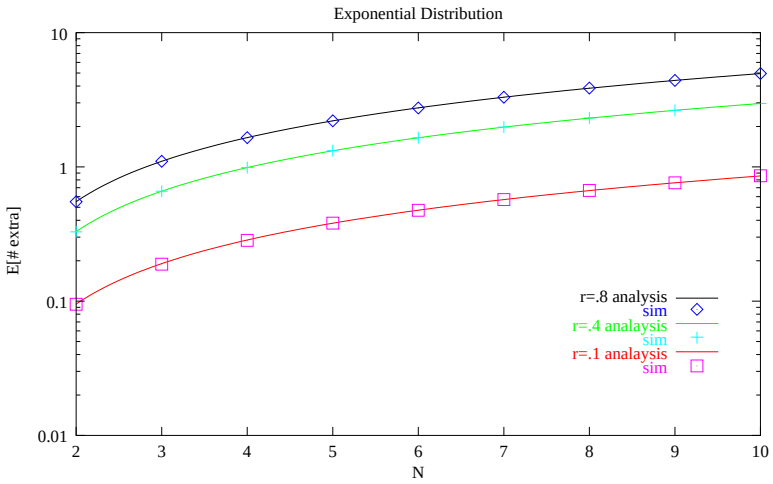

Figure 2.21: **Simulation vs. Analysis: Extra Messages vs. N (Large $r = \Delta/T$).**

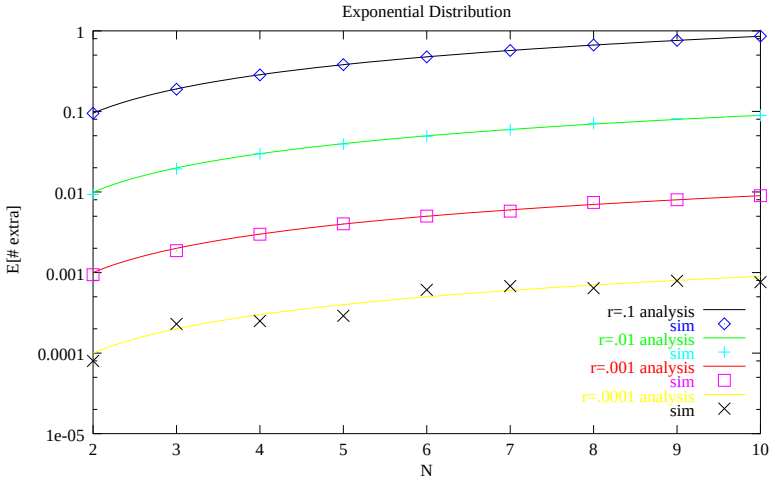

Figure 2.22: **Simulation vs. Analysis: Extra Messages vs. N (Small $r = \Delta/T$).**

2.7 Related Work

Suppression as a technique to delay messaging has its origin in the packet radio ALOHA protocol [1] and the Ethernet protocol [44]. The protocols that were designed and subsequently optimized for these media share an important characteristic with multicast communication. Namely, all messages sent to the group address are sent to all receivers subscribed to that address. Packet radio and Ethernet, however, are contention networks. Because the physical media are shared among all processes, there is the potential for messages to collide during transmission. Consequently, techniques like random delay are used to help manage bandwidth for collision avoidance and after collision detection (e.g., exponential backoff).

A key difference between Suppression as used with ALOHA versus with multicast is that it is employed after a collision is detected rather than at the beginning of a transmission. Ethernet,

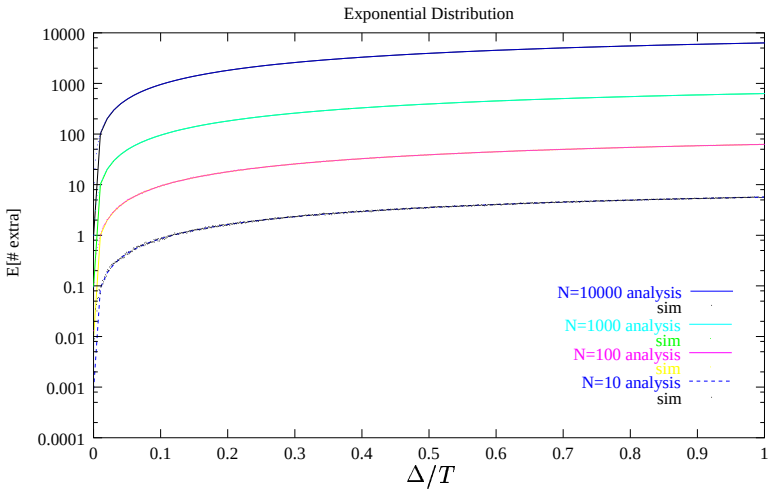

Figure 2.23: **Simulation vs. Analysis: Extra Messages vs. Δ/T (Exponential).**

on the other hand, does rely on processes to listen before sending (e.g., carrier-sense) and thus act accordingly (not send if another process is in the middle of transmitting and reschedule to re-sense the network in a random period of time). A notable attribute of Ethernet is that it can rely on the fact that transmission delay between processes is uniformly low. In fact, for Ethernet the propagation delay is assumed to be small compared to the transmission time to send the message. Propagation delay is also considered identical for all processes. As a result of the physical properties of the networks, the kinds of metrics of interest to Ethernet and ALOHA are the number of collisions generated, the backlog of processes waiting to retransmit, and the channel utilization.

Multicast as a communication technique is merely an abstraction on top of a variety of physical media. Therefore, it is not as controlled an environment and processes do not have the luxury of being able to sense that other processes are in the middle of transmissions. Therefore, its use of Suppression is as a first line of defense, rather than as an action taken after detecting a potential collision. Conversely, because a Multicast process cannot sense the current state of the other processes of the network, until such time as it receives a message, processes do not employ multiple rounds of Suppression for back-off purposes like in Ethernet. However, a strategy similar to back-off has been used by pairs of request-response Suppression algorithms; in contrast to Ethernet, where back-off is used to avoid collisions, it is used with Suppression to ensure successful retransmission of a repair request [27].

There is precedent in studying adaptive algorithms for Suppression. Floyd et al. in their study of SRM [27] propose an adaptive Suppression interval, T. As their work is aimed at Suppression for reliable multicast, there is both a request and a response phase, each of which relies on a separate but related Suppression algorithm with its own pair of timer values. The two phases are tightly coupled in their analysis and simulations. A major difficulty with their approach is that it relies on

each process to estimate the distance (the roundtrip delay) between every other process.

Kermode's work on SharqFEC [41] [42] attempts to make the SRM algorithm more scalable by partitioning the group of processes into administratively-scoped multicast zones. This produces more localized regions of control in which delays are estimated between zones, rather than between every pair of processes. Suppression plays a key role in these estimates and once again is employed to reduce message implosion.

The research by Nonnenmacher et al. is most similar to ours, in that they study a series of distributions from an analytic standpoint [45] [46] [47]. Considering their work as a starting point, we attempted to clarify some of their seminal analysis work. We were especially interested in their discovery of the positive, truncated exponential distribution, which has very promising performance characteristics under certain conditions. However, this distribution (and the focus of their work in general) is very firmly set in the context of reliable multicast data transfer. Thus the distribution is optimized for a specific operating point; up to 10^6 group members, minimizing the messaging overhead, and using Suppression in back-to-back mode (as mentioned above) for requesting data retransmission, as well as data replenishment. In contrast, we try to speak more generally about parameter interaction in order that the appropriate operating point can be derived for different applications and different operating environments.

In this work as well as that presented by Nonnenmacher, the uniform distribution is used as a baseline for comparison (since it is most commonly deployed). Yet each presents a different exponential algorithm as an alternative, highlighting the scenarios under which it outperforms the more traditional scheme. Our exponential function is of the form $Ae^{-\alpha t}$, whereas their exponential function is of the form $Ae^{\alpha t}$. The negative form of the function, which is not truncated, spreads the messages over a potentially larger time interval, attempting to avoid unnecessary messaging. The positive form of the function produces a small number of quick responders, with many more slow responders. This behavior is useful for applications that aim to avoid message implosion. As such, it has led us to begin examination of a more extreme example of this type of distribution, which we refer to as a "bin" approach and which we discuss in Section 2.9. The general idea is that the function has two or more distinct regions of operation. For example, with the function proposed by Nonnenmacher et al., in one region the function keeps $E[\#\ extra]$ constant, whereas in the other it grows linearly. The linear part of the function does not dominate until N is large, allowing for $E[\#\ extra]$ to remain small. Hypothetically, it should be possible to design a timer distribution function that optimizes the performance of a particular Suppression metric, given that we know the range of various operating parameters.

Minimizing $E[\#\ extra]$ is a compelling goal for reliable multicast. SRM [27] accomplishes this by using a uniform distribution and adjusting T during each round of Suppression. The work by Nonnemacher et al. [45] aims to eliminate the reliance on an accurate estimate of N, the number of

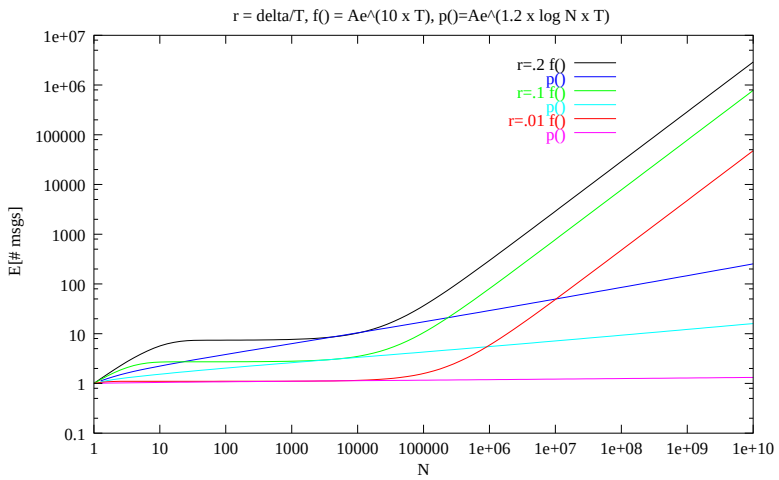

Figure 2.24: **Improvement on Positive, Truncated Exponential.**

participants, and the amount of state required at each receiver. Although the resulting solution of a positive, truncated exponential timer function is a useful alternative, it falls short of providing a general approach. First, the positive form of the function is optimized for a particular range of the operating space, and then only roughly. On closer examination of the α parameter in $Ae^{\alpha t}$, we found that their use of $\alpha = 10$ is only a rough estimate of an optimal value for the range $(10^0 - 10^6)$ over which their function is designed to work. In fact, $\alpha = 10$ is only optimal for 10^4 participants. There are better choices for α for different subranges within the larger range. Were there an application with a more specific operating range for N, the truncated, positive exponential distribution could be further improved by picking a different α. Second, most of the analysis of their function examines Δ/T as ranging from $[.1, 1]$. We have shown that there are important applications where the ratio falls within $[0, .1]$ and needs more complete investigation. Our work attempts to study the full range, or at very least, to study the boundary between large and small ratios. Finally, although their scheme is less reliant on group size N than other schemes, it is criticized in [50] because the timer distribution is actually tuned for an assumed range of N operation, and "once one has this information it might be better used in some local recovery approach rather than using it merely to tune the timer parameters."

This in turn leads to the question of designer functions in general. On cursory examination, we found that by setting $\alpha = 1.2 \times log\ N$, we could create a function that outperformed the original function with regards to $E[\#\ extra]$ (Figure 2.24). Due to the inherent property that there is a tradeoff between minimizing the metric for overhead versus the metric for latency, our designer function not surprisingly performed slightly worse (5%) for the metric $E[t_{min}]$.

2.8 Summary of Results

It is clear from the analysis that there exists an inverse relationship between the metric for messaging overhead, $E[\#\ extra]$, and for messaging delay, $E[t_{min}]$. Unfortunately, gains in one are offset by losses in the other. Hence, the optimal balance will be a function of the operating environment and the application employing the Suppression algorithm.

Our analysis showed and our simulations confirmed that Suppression may not be as effective for very large groups as it is for small or moderate-sized groups because it is difficult to suppress the numbers of extra messages. This is especially true when it becomes difficult to keep Δ/T ratio predictably low (i.e., in the wide-area). We observed that when N is very large, Δ/T must be extremely small to avoid extra messages. Conversely, when N is small, the ratio can afford to be larger.

If the goal is to keep $E[\#\ extra]$ below a given threshold, but N or Δ fluctuate, then T will need to adapt to compensate. Floyd et al [27] propose an adaptive scheme for reliable multicast that employs two back-to-back Suppression algorithms. This scenario has the benefit of a request-reply interaction to estimate the delay between pairs of processes and uses multiple polling rounds for accurate estimation. We corroborate their findings and differentiate our results in Section 2.7.

To our surprise, setting the metrics equal showed that the decaying exponential distribution outperforms the uniform distribution under a number of conditions. When the $E[\#\ extra]$ are equal and N is small, the exponential distribution has smaller elapsed delay $E[t_{min}]$, when the ratio of Δ/T is greater than .1, and with larger T. When $E[t_{min}]$ are equal, fewer extra messages $E[\#\ extra]$ are generated with the exponential distribution when the Δ/T ratio is large and more significantly as N increases.

Directly comparing the results of the uniform and decaying exponential probability density functions suggest that it may make sense to design systems that use one distribution in one part of the operating range, and adapt to use another distribution when in another part of the operating range. This is particularly interesting given that most deployed multicast algorithms using Suppression select random delay timer values from uniform distributions.

In summary, the work presented in this chapter is novel on several counts. Our analysis is aimed at understanding the core Suppression algorithm; a single iteration of it, completely decoupled from any previous or subsequent execution of the algorithm. Our work distinctly does not assume that it is used in conjunction with a second Suppression algorithm, nor that it is used iteratively. We acknowledge that any effective adaptation scheme will rely on multiple iterations in order to accrue history, but that past state information may be obtained through a number of means, some or all of which may be the repetition of the Suppression algorithm.

The study of Suppression as a stand-alone element allows us to understand the effect of composi-

tion, when Suppression is combined with other algorithms, including with the Suppression algorithm itself. Our focus has been to identify the parameters that play a role in the outcome of the metrics. Our work examines alternative distribution strategies than ones proposed and/or deployed, as well as presents the underlying theoretical basis for the performance of the algorithm as observed in operation or through simulation. Our study aims to understand the theoretical underpinnings, not of a single parameter, but of each network parameter in isolation, with a goal of improving the parameterization of deployed network protocols as well as those under design.

2.9 Future Work

There are numerous directions in which to take this research. We present a sampling of the possibilities in the sections below: we discuss several ways in which the model itself could be extended to be more sophisticated, we describe a rough approach for the derivation of unknown parameters from observed system behavior, and we propose additional Suppression distributions for investigation.

Refining the Model. Because we make no assumptions about the iterative use of the protocol, we assert that the delay between pairs of processes are fixed over the duration of the algorithm. However, if we relax this constraint, then we can more effectively study the iterative use of the algorithm and its combination with other algorithms (in order to collect past history), in order to support adaptive Suppression strategies.

We have assumed that the delay between pairs of processes is fixed. This assumption is only true under certain conditions, for example, in a local area network, where delay between pairs of processes is characterized as being a fixed Δ within a small amount of variance. The larger the region encompassed topologically by a set of processes, the less valid a fixed-Δ model is likely to be. However, it may still be acceptable to describe a set of processes as having an *effective* Δ, when the set behaves as if Δ were a particular value on average. In particular, we would like to explore three questions regarding transmission delay variance of Δ:

- What are the effects of an *outlier* process, one that has transmission delay considerably larger or smaller than the rest of the group members?
- By how much can the delay vary between pairs of processes before our fixed-Δ analysis is no longer applicable?
- What are the effects of multiple classes of Δ on the behavior of the algorithm, i.e., when there exist subsets of processes within discrete ranges of delay from each other?

In this thesis, we also assume that delays between pairs of processes do not vary over time, as might happen under light versus heavy loads. Our reasoning is that the Suppression algorithm, when used

in isolation (versus iteratively) is typically short-lived, meaning that T, the upper bound on the delay interval, is small. In this case, we believe it is still useful to employ the algorithm and to parameterize it to optimize the average expected behavior.

Because Suppression is often employed in combination with other algorithms, many of which perform process synchronization before relying on Suppression, we have deferred modeling asynchronous process arrivals in this chapter. Our rationalization has been that Suppression is often targeted to solve implosion when processes arrive at approximately the same moment. Synchronized process arrivals might occur after a system reboot, after the explicit receipt of a message or at the beginning of a scheduled teleconference. On the other hand, asynchronous process arrivals might occur as group membership fluctuates. The questions then become, when processes arrive randomly into the system, what is the impact on the Suppression metrics proposed, i.e., is Suppression still an effective technique for scalability? While Floyd et al. touch upon this issue when they study the impact of topology on back-to-back Suppression algorithms [27], we would like to reframe the question in terms of delay variance, as applied to the basic Suppression algorithm.

Derivation of N and Other Parameters. It can be difficult to ascertain the group size of a set of communicating processes [51]. If the group size is large, an active method like querying all group members is untenable because of the potential for message implosion. A passive method, such as simply listening and counting the number of processes from which messages are received, is only useful when all processes issue messages (as with Announce-Listen). With the Suppression algorithm, the expectation is that only a very small number of processes will actively issue messages.

Yet the estimation of group size is important for scaling multi-process communication. It is often the case that N is derived through a combination of listening and approximation techniques [30] [29]. Once derived, N is used to bound the amount of bandwidth dedicated to control messaging. The difficulty is ascertaining an accurate estimation quickly, for example immediately at startup or after a large membership fluctuation occurs [52].

In [46], Nonnenmacher suggests using Suppression variables to derive a first approximation for group size estimation in a single round. If the participating processes agree on a set timer distribution, then we can derive N from the latency and overhead characteristics. For example, if the first message is received at $t_{recv} = t_{min} + \Delta$, and all processes use a uniform pdf with a fixed T and Δ, then $N = T/(t_{recv} - \Delta) - 1$. Nonnenmacher actually uses the positive, truncated exponential distribution and assumes processes synchronization through the receipt of a prior message. In [30] [29], Friedman et al. refine the technique, but note that the original distribution and Suppression algorithm lead to bias; above N for low values of N, and below N for higher values. This may be in part due to the usage of an α value that has been optimized for 10^4 participants. It would be interesting to compare the behavior of other distributions for group size estimation and combine them with

the improvements cited in [30] [29] (notably refinements to make them robust even when there are heterogeneous delays between processes).

Once N is derived, a process can determine which part of the operating space it is in and could switch between appropriate pdfs, for example, using decaying exponential distribution for small N, and the uniform distribution for large N. Along similar lines, if we obtain N, and know T, then we could derive an estimate of average transmission delay, Δ, by obtaining the number of processes which issue Suppression messages. This might be particularly useful when there are heterogeneous delays between participating group members, but the delay variance is within some bound. The idea would be to use this information to create administratively-scoped multicast groups for groups of processes with similar delay characteristics.

"Two-Component" Uniform Distribution. The realization that uniform and exponential distributions outperform each other in different parts of the parameter space raises the larger question of whether there are other probability distribution functions in need of exploration. We have suggested that systems could benefit by adapting to use one pdf in one part of the parameter space and another outside that range. Another strategy would be to use different pdfs for different classes of processes. For instance, such a strategy might be advantageous when processes can be easily categorized into server vs. client, router vs. end system, high-speed vs. low-speed (link, cpu, memory), leader vs. non-leader. Below, we introduce the two-component uniform distribution as an example of the kind of distribution that can be designed to differentiate between processor classes.

The inspiration behind this distribution is Nonnenmacher's exponential timer distribution function [45], which leads to two operating regions for number of extra messages generated. In one region the function behaves linearly and in the other it remains constant. A generalization of this approach would be to create a small number of regions of the function so different classes of processors would behave differently to the wake-up timer.

The two-component uniform distribution is like a uniform distribution except the pdf is zero for $a \leq t < a + w$. The two-component uniform distribution is the base case for the N-component uniform distribution, the separation of processes into N "bins" of equal width and spacing.

The motivation behind exploring a *bins* theory is that there may be some benefit to separating the group of processes into 2 or more classes. Certain processes will respond quickly and others more slowly, i.e., processes nearby vs. further away, or processes with more CPU power vs. those without. Therefore, the bins serve to distinguish between the different processor classes. The width of the bins and the space between them is critical to spread N processes into the proper bins and ultimately to help reduce the number of processes issuing messages. Multiple bins could be implemented by rounding up or down, so that the value chosen from the uniform distribution falls within the range of one of the bins.

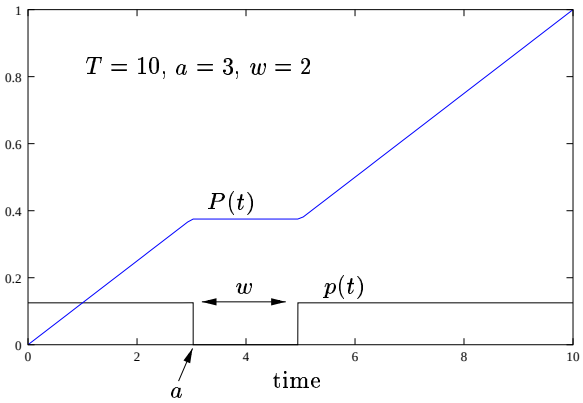

Figure 2.25: **Two-Component Uniform Distribution.**

Due to the complexity of an N-bin distribution, we outline an approach for studying the simplest form of the algorithm when $N = 2$. The two-component uniform distribution has three disjoint ranges. We derive $E[t_{min}]$ by splitting the integral into these three pieces and substituting $P(t)$ for each range.

$$p(t) = \begin{cases} 1/(T-w) & \text{for } 0 \le t < a \\ 0 & \text{for } a \le t < a+w \\ 1/(T-w) & \text{for } a+w \le t < T \end{cases}$$

and

$$P(t) = \begin{cases} t/(T-w) & \text{for } 0 \le t < a \\ a/(T-w) & \text{for } a \le t < a+w \\ (t-w)/(T-w) & \text{for } a+w \le t < T \end{cases}$$

$$\begin{aligned} E[t_{min}] &= \int_0^T (1-P(t))^N \, dt \\ &= \int_0^a (1-t/(T-w))^N \, dt + \int_a^{a+w} (1-a/(T-w))^N \, dt \\ &\quad + \int_{a+w}^T (1-(t-w)/(T-w))^N \, dt \\ &= \frac{T-w}{N+1} + w(1-\frac{a}{T-w})^N \end{aligned}$$

For the rest of the calculation, we assume that $w \ge \Delta$, $\Delta < a$, and $\Delta < T - a - w$. The first assumption corresponds to the case when the width w of the "hole" in the pdf is greater than or equal to Δ. The second and third assumptions mean that the width of the two ranges where the

pdf is non-zero are both larger than Δ. We make these assumptions to simplify the integrals.

$$\begin{aligned}
E[\#\ extra] &= N \cdot P(\Delta) - 1 + N \int_{\Delta}^{T} p(t)(1 - P(t-\Delta))^{N-1}\, dt \\
&= N \cdot P(\Delta) - 1 + N \int_{\Delta}^{a} p(t)(1 - P(t-\Delta))^{N-1}\, dt + \\
&\quad N \int_{a}^{a+w} p(t)(1 - P(t-\Delta))^{N-1}\, dt + \\
&\quad N \int_{a+w}^{a+w+\Delta} p(t)(1 - P(t-\Delta))^{N-1}\, dt + \\
&\quad N \int_{a+w+\Delta}^{T} p(t)(1 - P(t-\Delta))^{N-1}\, dt \\
&= \frac{N\Delta}{T-w} - \left(1 - \frac{a-\Delta}{T-w}\right)^{N} + \frac{N\Delta}{T-w}\left(1 - \frac{a}{T-w}\right)^{N-1} + \left(1 - \frac{a+\Delta}{T-w}\right)^{N}
\end{aligned}$$

To get a feel for this expression, let $\alpha = 1 - \frac{a}{T-w}$. We have already assumed $\Delta < T - w - a$. If we further assume that $\Delta \ll T - w - a$,

$$\begin{aligned}
E[\#\ extra] &= \frac{N\Delta}{T-w} + \alpha^{N}\left(-\left(1 + \frac{\Delta}{T-w-a}\right)^{N} + \frac{N\Delta}{T-w-a} + \left(1 - \frac{\Delta}{T-w-a}\right)^{N}\right) \\
&\leq \frac{N\Delta}{T-w} - \alpha^{N}\frac{N\Delta}{T-w-a}
\end{aligned}$$

An interesting case is when the gap in the pdf is symmetric, i.e., $a = (T-w)/2$. In this case, $\alpha = 1/2$. Note that for $E[t_{min}]$, the value in parentheses is always non-negative.

$$\begin{aligned}
E[t_{min}] &= \frac{T-w}{N+1} + \frac{w}{2^N} = \frac{T}{N+1} - w\left(\frac{1}{N+1} - \frac{1}{2^N}\right) \\
E[\#\ extra] &\leq \frac{N\Delta}{T-w} - \frac{1}{2^N}\frac{2N\Delta}{T-w} = \frac{N\Delta}{T-w}\left(1 - \frac{1}{2^{N-1}}\right)
\end{aligned}$$