Chapter 3

Suppression with Loss

In this chapter we examine the impact of loss on the model for Suppression that we introduced in Chapter 2. We revisit the metrics defined earlier but re-evaluate them under lossy conditions. In addition, we present several new metrics to predict performance when loss occurs. These include metrics for the effective completion time for the algorithm (Maximum Time Elapsed); the total number of messages generated during the normal course of the algorithm (Number of Messages Generated); and an estimate of how many of the messages that are sent are really necessary (Messages Required) and how many are overhead (Extra Messages with Loss). We discuss several other metrics that impact CPU performance, which are related to the probability of message receipt.

In addition to studying appropriate metrics, this chapter studies the effects of both fully correlated and fully uncorrelated loss. Fully correlated loss occurs when a message is lost closest to the sender, so its loss is experienced by all receivers. With fully uncorrelated loss, a message is lost closest to the receivers, thus the losses experienced by each of the receivers may be different. Most multicast groups will experience loss somewhere between uncorrelated and correlated behavior, so we calculate both as a way to bound the best and worst behavior. Throughout this chapter and the remainder of the thesis, we use the terms correlated and uncorrelated to mean fully correlated and fully uncorrelated, respectively.

Before concluding, we provide an overview of related work, summarize our results, and propose future directions for our research.

3.1 $E[t_{min}]$ Re-visited: Time Elapsed with Loss

Because $E[t_{min}]$ was defined as the earliest time that a message is sent, packet loss does not actually have any impact on the earliest possible time that a node *selects* a suppression wake-up time. However, the message sent by the process that selects the earliest time might be lost by the network. Therefore, different processes participating in the Suppression (SUP) algorithm might be suppressed by different messages. For Suppression with loss, we therefore define $E[t_{min_e}]$, an "effective" $E[t_{min}]$,

that we use for comparisons. $E[t_{min_e}]$ is the expected time of the earliest message sent but *not completely dropped* in the network. Therefore, $E[t_{min_e}]$ represents the earliest hope for suppression by a remote process.

Given a particular vector of times $\vec{t} = (t_0, t_1, \ldots, t_{N-1})$, where each t_i is the wake-up time selected by the individual processes, let $\vec{T}_{min} = (t_{min_0}, t_{min_1}, t_{min_2}, \ldots, t_{min_k}, \ldots, t_{min_{N-1}})$ be a permutation of the vector $\vec{t}$ such that $t_{min_i} \leq t_{min_{i+1}}$ for $0 \leq i < N-1$. Thus, $\vec{T}_{min}$ is a function of the t_i sorted in non-decreasing order. Note that the value t_{min_0} is the same as t_{min} (defined in Chapter 2, Section 2.3.1) and $t_{min_{N-1}}$ is the largest time selected by any of the processes.

In Figure 3.1, we display a Suppression interval of length T. Each of N processes selects a time to awaken, t_i, and these are shown ordered from t_{min_0} to $t_{min_{N-1}}$. Under lossy conditions, some subset of the processes will awaken to find that they have not received a message from any other process; these processes, indicated with an `x`, generate a message. The remaining processes, indicated with an `o`, are suppressed.

If messages can be lost, then processes might be suppressed by a process other than the one awakening at t_{min_0}. Furthermore, each process may be suppressed by a different t_{min_i}, since a message received by one process may be lost while in transit to another.

3.1.1 Loss Analysis

We examined both correlated and uncorrelated loss. Simulations showed that correlated loss produces $E[t_{min_e}]$ that is higher than the uncorrelated case. Therefore, we present a detailed analysis of the correlated case as it bounds the performance of the uncorrelated case and the Suppression algorithm in general.

Correlated Loss. Let the probability of message loss be given by l. If the message sent at the earliest selected time t_{min_0} arrives successfully, it represents the earliest message to arrive at process i that might suppress it. Successful arrival happens with probability $(1-l)$. If instead the earliest message is lost, we examine the message sent by the process with the next earliest time t_{min_1}. If that message is lost, we examine the next time selected t_{min_2}, and so forth. We assume that not all messages are lost. Thus, the expected time of the earliest message sent to process i taking loss into account is given below. The normalization factor $1/(1-l^N)$ comes from the assumption that not all messages are lost.

$$E[t_{min_e}] = E\left[\frac{((1-l)t_{min_0} + (1-l)lt_{min_1} + \cdots + (1-l)l^k t_{min_k} + \cdots + (1-l)l^{N-1}t_{min_{N-1}})}{(1-l^N)}\right]$$
$$= \frac{(1-l)}{(1-l^N)} E\left[\sum_{0 \le k < N} l^k \cdot t_{min_k}\right]$$

We need to calculate $E[t_{min_k}]$, the expected value of the kth smallest time given a randomly chosen vector of N times. For a particular time t_i to be the kth smallest, there must be exactly k times smaller than it, and $N-1-k$ times larger than it (note that indices begin at 0). The probability of this event is $\binom{N-1}{k} P(t_i)^k (1-P(t_i))^{N-1-k}$. Therefore, the expected value is given by:

$$E[t_{min_k}] = \int_0^T Ntp(t)\binom{N-1}{k} P(t)^k (1-P(t))^{N-k-1}\, dt$$

Substituting $E[t_{min_k}]$ into the equation for $E[t_{min_e}]$, we derive the formula below. We use the binomial expansion to simplify the equation and let $Q(t) = (1-l)P(t), q(t) = (1-l)p(t)$, where $q(t) = dQ/dt$. Note that the resulting formula is similar in form to the expression for Suppression without loss, and when $l = 0$ the formula is identical to the expression derived in Section 2.3.1.

$$E[t_{min_e}] = \frac{(1-l)}{(1-l^N)} \sum_{0 \le k < N} l^k \int_0^T Ntp(t)\binom{N-1}{k} P(t)^k (1-P(t))^{N-k-1}\, dt$$
$$= \frac{(1-l)}{(1-l^N)} \int_0^T Ntp(t)(lP(t) + (1-P(t)))^{N-1}\, dt$$
$$= \frac{-Tl^N}{(1-l^N)} + \frac{1}{(1-l^N)} \int_0^T (1-Q(t))^N\, dt$$
$$= \frac{-Tl^N + \int_0^T (1-Q(t))^N\, dt}{(1-l^N)}$$

3.2 Maximum Time Elapsed

It is common for the Suppression algorithm to be characterized in terms of $E[t_{min}]$, i.e., when the algorithm sends its first message. However, it is equally useful, particularly with message loss, to ask how long it takes for the algorithm to complete. Thus we define $E[t_{max}]$, when the algorithm sends its last message. Completion time is important when Suppression is followed by another algorithm, which is often the case.

The value $E[t_{max}]$ is defined as the expected time selected by the last process that actually generates a message (Figure 3.1). This is appreciably different than the maximum t_i selected by all processes. In other words, $t_{max} \neq t_{min_{N-1}}$, and when $l \neq 1$ we would expect that $t_{max} < t_{min_{N-1}}$.

In the lossless case, $E[t_{min}] \le E[t_{max}] \le E[t_{min}] + \Delta$, meaning the last message is sent within

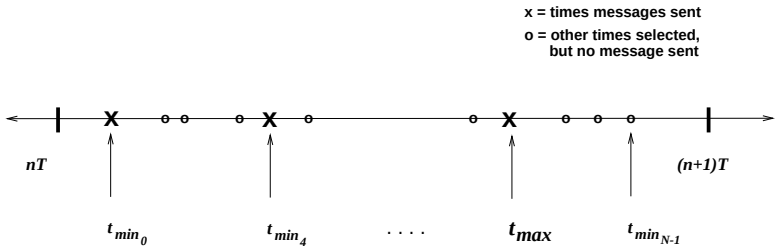

Figure 3.1: t_{max}: **The Maximum** t_{min_i} **Sent.**

Δ of the earliest message. However, with loss, multiple messages may be necessary to suppress a group of processes. Thus, the time of the last message sent, $E[t_{max}]$, is no longer defined in terms of the time of the earliest message sent, $E[t_{min}]$. In either case, $E[t_{max}] + \Delta$ can be thought of as delineating a point in time beyond which all processes are considered suppressed, i.e., the expected time after which all nodes are in agreement to halt the algorithm.

3.2.1 General Form

Consider $Pr[t_{max} = t_{min_k}]$, the probability that the maximum time elapsed equals the k^{th} value in the vector $\vec{T}_{min}$. The likelihood that t_{min_k} is the last suppression message actually sent is the probability that the k^{th} process does not receive a message from any earlier processes and that all later processes receive at least one message from one of the earlier processes or the k^{th} process. A process k may not receive a message from another process for one of several reasons:

- the other process never sent a message because it was suppressed,
- the message was sent but was lost, or
- because the message arrived late (it was generated after $t_{min_k} - \Delta$).

In general, we can write

$$E[t_{max}] \quad = \quad \sum_{0 \leq k < N} t_{min_k} \times Pr[t_{max} = t_{min_k}]$$

3.2.2 Zero Delay

The calculation of $E[t_{max}]$ is subtle since the event $(t_{max} = t_{min_k})$ consists of two parts that are interdependent: process k must lose all messages sent before it, and every process after k must not lose all the previous messages from processes $\{0, \ldots, k\}$. These parts are dependent because they both rely on particular messages actually being sent (or not) by processes $\{0, \ldots, k-1\}$.

Furthermore, the calculation of $E[t_{max}]$ is complicated by the combined effects of both message loss and delay. We attempt to differentiate between their effects on $E[t_{max}]$ by understanding how

Event	*Probability*
$t_{max} = t_0$	$(1-l)^2$
$t_{max} = t_1$	$l(1-l^2)$
$t_{max} = t_2$	$l((1-l)+l^2)$

Table 3.1: **Zero-Delay Event Probabilities.**

the system behaves when each is set to 0. In Chapter 2, we studied Suppression with zero loss. Here we isolate the effect of loss by setting the transmission delay to as close to zero as permitted by the network simulator.[1] The result of setting Δ as close to zero as possible (and also very small relative to T) is that communication with other processes happens virtually instantaneously. This makes it statistically improbable that a message is generated within Δ of other messages and thus we can observe the impact of loss on the algorithm.

Consider a three node example that illustrates the difficulties of modeling $Pr[t_{max} = t_{min_k}]$. For simplicity let us assume that $\vec{T}_{min} = \vec{t}$, that the processes $0, 1, 2$ select wake-up times t_i in ascending order. Let us also assume that there is zero delay in the network.

$\mathbf{t_{max} = t_0}$**.** Process 0 generates the last message sent if message 0 was received by processes 1 and 2. Each of these events occurs with probability $(1-l)$. Therefore, $Pr[t_{max} = t_0] = (1-l)^2$.

$\mathbf{t_{max} = t_1}$**.** Process 1 generates the last message sent if message 0 was not received by process 1 (which happens with probability l), and either message 0 was received by process 2 or message 0 was lost by process 2 and message 1 was received by process 2 (which is equivalent to saying process 2 did not lose both messages sent, $(1-l^2)$). Therefore, $Pr[t_{max} = t_1] = l(1-l^2)$.

$\mathbf{t_{max} = t_2}$**.** Process 2 generates the last message if process 2 lost message 0 (occurring with probability l), and process 1 never sent a message (because it received the message from process 0, with probability $(1-l)$) or process 1 sent a message (because it lost the message from process 0, with probability l) and it was lost by process 2 (which occurs with probability l). Therefore, $Pr[t_{max} = t_2] = l((1-l)+l^2))$.

The probabilities for these events are summarized in Table 3.1. The example highlights that we must take into account that some messages are *never sent due to having been suppressed previously!* In contrast, the process that selects the minimum time *always* sends its message. We must be careful not to assume when process k loses all previous messages that all k messages were actually generated; some of them may have been suppressed by other messages. In fact, there will be $1 < i \leq k$ messages generated. Moreover, when we consider that all processes beyond process k do not lose

[1] It is impossible in `ns` to have zero delay, so we characterize it as approximately zero. Although we set link delay to 0, there is a small amount of time consumed for switching delay through nodes. We offset this by making the T interval sufficiently large to mask the effects.

all the messages sent, we must be careful not to double count the loss by assuming that all $k+1$ messages were generated.

Simulation results showed that uncorrelated loss produces $E[t_{max}]$ that is higher than the correlated case. Therefore, we present a detailed analysis of the uncorrelated case as it serves as an upper bound on the performance of the algorithm.

Uncorrelated Loss. Let us derive $Pr[t_{max} = t_{min_k}]$ more precisely in the case where there is zero transmission delay in the network. t_{max} is equal to t_{min_k} when process k loses all messages that were sent from $0 \leq j < k$, and processes $k+1 \leq j < N-1$ receive at least one message from processes $0 \leq j \leq k$ that generate messages (Figure 3.2). This can be decomposed into a set of disjoint events, each event being the case when processes in the range $0 \leq j < k$ generate i messages, where i ranges from 1 to k.

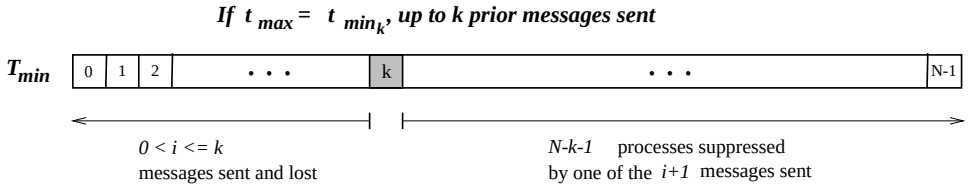

Figure 3.2: $Pr[t_{max} = t_{min_k},\ given\ i\ messages\ sent]$.

Given that exactly i messages were sent, process k must have lost i messages, which occurs with probability l^i. Additionally, all $N-k-1$ processes with later wake-up times than process k each must have been suppressed by an earlier process, meaning that each did not lose all of the $i+1$ previous messages sent. Also, the probability that $t_{max} = t_{min_0} = t_{min}$ is equal to the likelihood that all subsequent $N-1$ processes receive the first message sent. These observations are summarized by the following equations:

$$Pr[t_{max} = t_{min_k}]_{\Delta=0} = \sum_{1 \leq i \leq k} Pr[exactly\ i\ messages\ sent] \times l^i (1 - l^{i+1})^{N-k-1}$$

$$Pr[t_{max} = t_{min_0}]_{\Delta=0} = (1-l)^{N-1}$$

We now determine the probability that exactly i messages are sent by processes 0 to $k-1$. Let $P(i,n)$ be the probability that exactly i messages are sent by n processes. Note that the probability $Pr[exactly\ i\ messages\ sent]$ is simply $P(i,k)$.

$P(i,n)$ can be broken down into two disjoint parts: (a) Process $n-1$ sends a message, and processes $0 \leq j < n-1$ send $(i-1)$ messages; (b) Process $n-1$ does not send a message, and processes $0 \leq j < n-1$ send i messages. In case (a), process $n-1$ must lose the $(i-1)$ messages

sent, which happens with probability l^{i-1}. In case (b), process $n-1$ must receive at least one of the i messages sent, which happens with probability $1-l^i$. Therefore, we can write:

$$\begin{aligned} P(i,n) &= Pr[\textit{exactly } i \textit{ messages sent by } n \textit{ processes}] \\ &= P(i-1,n-1) \times l^{i-1} + P(i,n-1) \times (1-l^i) \end{aligned}$$

Since the process with the earliest selected time always sends a message, $P(0,n) = 0$ for $n > 0$. We can also compute $P(i,i)$, because this is the likelihood that each process sends a message, i.e., each process loses all messages sent by any earlier process. Therefore,

$$\begin{aligned} P(0,n) &= 0 \\ P(i,i) &= l^1 \cdot l^2 \cdot l^3 \cdots l^{i-1} \\ &= l^{(i-1)i/2} \end{aligned}$$

Thus,

$$\begin{aligned} E[t_{max}]_{\Delta=0} &= \sum_{0 \le k < N} t_{min_k} \times Pr[t_{max} = t_{min_k}]_{\Delta=0} \\ &= \sum_{0 \le k < N} t_{min_k} \times \sum_{0 \le i \le k} Pr[\textit{exactly } i \textit{ messages sent}] \times l^i (1-l^{i+1})^{N-k-1} \\ &= \sum_{0 \le k < N} \int_0^T Ntp(t) \binom{N-1}{k} P(t)^k (1-P(t))^{N-k-1}\, dt \times \\ &\quad \sum_{0 \le i \le k} P(i,k) \times l^i (1-l^{i+1})^{N-k-1} \end{aligned}$$

This formula matches our intuition. When there is no message loss in the network ($l = 0$), $E[t_{max}]_{\Delta=0} = t_{min_0}$, and when all messages are dropped ($l = 1$), $E[t_{max}]_{\Delta=0} = t_{min_{N-1}}$.

3.3 Number of Messages Generated

We define the metric $E[\#\ messages]$ as the total number of messages that are generated by the algorithm. In the lossless case, this metric is simply $E[\#\ messages] = E[\#\ extra] + 1$, as the algorithm generates only one useful message and the rest extra messages. Given the derivation for $E[\#\ extra]$ from Chapter 2, $E[\#\ messages]$ in the lossless case is:

$$\begin{aligned} E[\#\ messages]_{l=0} &= 1 + E[\#\ extra]_{l=0} \\ &= N \cdot P(\Delta) + N \int_\Delta^T p(t)(1-P(t-\Delta))^{N-1}\, dt \end{aligned}$$

In the lossy case, multiple messages may perform suppression by affecting different subgroups of processes, thus we need to revise how $E[\#\ messages]$ is calculated. If we can determine the probability that i messages are sent, then:

$$E[\#\ messages] \quad = \quad \sum_{1 \leq i \leq N} i \times Pr[exactly\ i\ messages\ sent]$$

Uncorrelated Loss. Simulations conclusively show that uncorrelated loss leads to higher message generation. Thus we present an analysis of the expected number of messages generated in the uncorrelated case. From Section 3.2.2, we know the probability that i messages are sent in the zero-delay case, and therefore $E[\#\ messages]_{\Delta=0}$ is:

$$\begin{aligned} E[\#\ messages]_{\Delta=0} &= \sum_{1 \leq i \leq N} i \times Pr[exactly\ i\ messages\ sent] \\ &= \sum_{1 \leq i \leq N} i \times P(i, N) \end{aligned}$$

Although we do not present an analytic solution for $E[\#\ messages]$, we offer $E[\#\ messages]_{\Delta=0}$ as an approximation.

3.4 Number of Messages Required

We also define the average number of useful messages, $E[\#\ required]$. We can think of $E[\#\ required]$ as being the number of messages required to fully suppress a group of size N. This is equivalent to the number of messages generated when there exists no transmission delay. This assertion holds true because if all messages are received instantaneously, there are **no** extra messages generated due to transmission delay and all messages are therefore necessary for the Suppression algorithm to work properly. The definition for $E[\#\ required]_{l=p,\Delta=d}$ becomes:

$$E[\#\ required]_{l=p,\Delta=d} \quad = \quad E[\#\ messages]_{l=p,\Delta=0}$$

The implication is that regardless of the value of Δ, the number of required messages remains constant for a given level of loss.

Uncorrelated Loss. Uncorrelated loss leads to higher numbers of required messages. The analysis for $E[\#\ required]$ is identical to and follows from the previous section:

$$\begin{aligned} E[\#\ required]_{l=p,\Delta=d} &= E[\#\ messages]_{l=p,\Delta=0} \\ &= \sum_{1 \leq i \leq N} i \times Pr[exactly\ i\ messages\ sent] \end{aligned}$$

$$= \sum_{1 \le i \le N} i \times P(i, N)$$

3.5 $E[\#\ extra]$ Re-visited: Extra Messages with Loss

Whereas $E[\#\ messages]$ characterizes the network traffic and $E[\#\ required]$ the necessary traffic, $E[\#\ extra]$ can be thought of as network overhead in the system.

When there is no transmission delay and no loss, $E[\#\ extra]$ equals zero and only the earliest message is generated. When loss is present, but there is no delay, $E[\#\ extra]$ still equals zero; we know from Section 3.4 that all messages generated are required messages. When delay is introduced between processes, but there is no message loss, $E[\#\ extra]$ is calculated as the number of processes for which t_i falls within Δ of $E[t_{min}]$ (Chapter 2). The first message sent is "useful" in that it is the only one that suppresses processes in the system; the remaining messages are redundant. These extra messages result due to transmission delay and are mistakenly sent due to an earlier message not arriving in time.

With delay and loss present, the calculation of $E[\#\ extra]$ is more subtle. $E[\#\ extra]$ still reflects the number of messages that are sent due to transmission delay of the earliest message sent. However, it also captures the number of messages that are within Δ of any other processes generating messages, rather than strictly within Δ of the minimum time selected. Thus, the interplay between transmission delay and loss contributes to $E[\#\ extra]$, as the impact of Δ and l are no longer separable. For example, a message generated by a process is considered extra if it is received but does not suppress any other process **and** if after it is sent the local process receives a message with an earlier t_i.

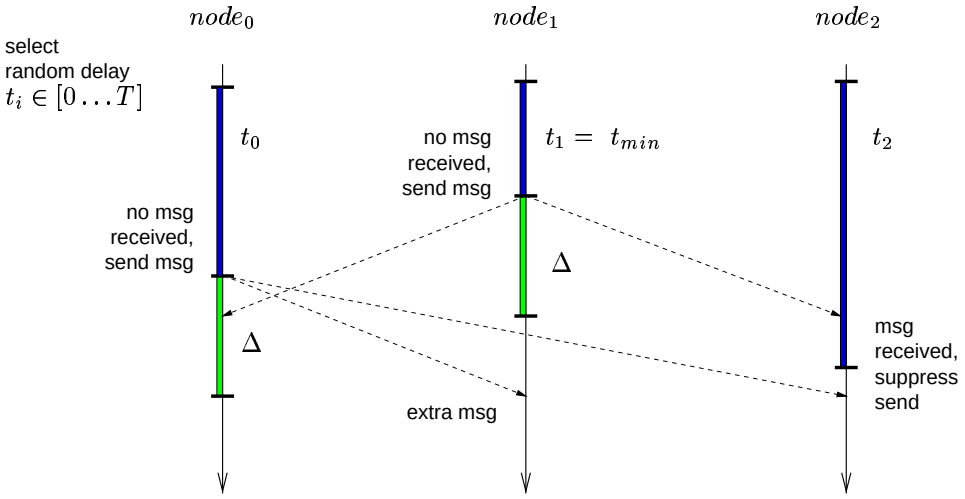

Figure 3.3: **Suppression Algorithm.**

For example, consider the three processes depicted in Figure 3.3. Assume that $T_{min} = (t_1, t_0, t_2)$.

If t_1 is within Δ of t_0, then process 0 does not receive message 1 in time to avoid sending its message. If process 1 actually receives the message from process 0, then message 0 is considered an extra message – because process 0 does not suppress other processes and it eventually receives an earlier message than its own. However, if the message from process 1 was dropped before it reached process 0, then message 0 is **not** considered extra. Note that the number of messages is a count of the extra messages sent (vs. received). Although there are N messages received, there is only 1 message generated. Copies are made at the branching points in the network.

Correlated Loss. Simulations show that correlated loss leads to many more extra messages than uncorrelated loss. This is in part a consequence of uncorrelated loss resulting in higher $E[\#\ messages]$ and $E[\#\ required]$. It is also due to the fact that for correlated loss, all messages up to and including the first successfully delivered message are considered required (as all processes lose or receive the same messages), and any successfully delivered messages within Δ of the first successfully delivered message are classified as extra.

Because of the complexity of deriving $E[\#\ extra]$, we simply relate it to $E[\#\ messages]$. We can determine the impact of a given transmission delay, $\Delta = d$, on the algorithm overhead, by subtracting the number of required messages generated when $\Delta = d$ from the total number of messages generated when $\Delta = d$. We assume all other parameters are kept constant.

$$\begin{aligned} E[\#\ extra]_{l=p,\Delta=d} &= E[\#\ messages]_{l=p,\Delta=d} - E[\#\ required]_{l=p,\Delta=d} \\ &= E[\#\ messages]_{l=p,\Delta=d} - E[\#\ messages]_{l=p,\Delta=0} \end{aligned}$$

We have already derived $E[\#\ messages]_{l=p,\Delta=0}$ analytically in the uncorrelated case. Here we present the analysis for the correlated case:

$$\begin{aligned} E[\#\ required] &= \sum_{1\le k\le N} k \times Pr[k\ messages\ sent] \\ &= \sum_{1\le k\le N} k \times (1-l) l^{k-1} \\ &= (1-l) \sum_{1\le k\le N} k \times l^{k-1} \\ &= \frac{l(1-l^N)}{(1-l)} - N\ l^N \end{aligned}$$

In other words, the probability that k messages are sent is the likelihood that $k-1$ messages were lost and the k^{th} message was successfully received.

As we have not derived an expression for the more general $E[\#\ messages]_{l=p,\Delta=d}$, to find $E[\#\ extra]_{\Delta=d}$, we subtract the analytic results for $E[\#\ required]_{\Delta=d}$ from the simulated results for $E[\#\ messages]_{\Delta=d}$.

3.6 Other Metrics

There are two related metrics that measure the impact of the reception of messages, but which we do not model here. The first is $E[\#\ messages\ received]$, the number of messages actually *received* on average. With message loss, the number received may be less than the number generated or sent. In the correlated loss case, all receivers will experience the same number of messages received. However, in the uncorrelated loss case, $E[\#\ messages\ received]$ may be different on a per node basis. Therefore, it becomes an averaged quantity. In effect, $E[\#\ messages\ received]$ gauges CPU resources used, as it represents the average number of interrupts a node receives. We also define $E[\#\ extra\ received]$, the number of extra messages received, which measures the CPU processing overhead experienced by the system. It too becomes an averaged metric in the uncorrelated loss scenario. On a per node basis, $E[\#\ extra\ received] = E[\#\ messages\ received] - 1$, assuming that one's own Suppression message is included in the number of messages received.

3.7 Distributions

Below, we examine the following probability distribution functions: a uniform distribution, and a decaying exponential distribution. We calculate the bounds for metrics derived in the previous sections of this chapter: $E[t_{min_e}]$, the effective minimum delay; $E[t_{min_k}]$, the kth smallest wake-up time; $E[t_{max}]_{\Delta=0}$, the maximum time elapsed; and $E[\#\ messages]_{\Delta=0}$, the total number of messages generated.

3.7.1 Uniform Distribution

For this case, $p(t) = 1/T$, and $P(t) = t/T$. Let $c = (1-l)$ and $Q(t) = cP(t)$. Then $q(t) = cp(t)$ and $q(t) = dQ/dt$.

$$\begin{aligned} E[t_{min_e}] &= \frac{-Tl^N + \int_0^T (1-Q(t))^N\,dt}{(1-l^N)} \\ &= \frac{-Tl^N + \int_0^T (1-cP(t))^N\,dt}{(1-l^N)} \\ &= \frac{T}{(N+1)}\left[\frac{1}{(1-l^N)}\left(\frac{(1-l^{N+1})}{(1-l)} - (N+1)(l^N)\right)\right] \end{aligned}$$

$$\begin{aligned} E[t_{min_k}] &= \int_0^T Ntp(t)\binom{N-1}{k}P(t)^k\left(1-\left(\frac{t}{T}\right)\right)^{N-k-1}dt \\ &= \int_0^T N\binom{N-1}{k}\left(\frac{t}{T}\right)^{k+1}\left(1-\left(\frac{t}{T}\right)\right)^{N-k-1}dt \end{aligned}$$

$$= \frac{T(k+1)}{(N+1)}$$

$$\begin{aligned} E[t_{max}]_{\Delta=0} &= \sum_{0\le k<N} t_{min_k} \times Pr[t_{max} = t_{min_k}]_{\Delta=0} \\ &= \sum_{0\le k<N} t_{min_k} \times \sum_{0\le i\le k} Pr[exactly\ i\ messages\ sent] \times l^i(1-l^{i+1})^{N-k-1} \\ &= \sum_{0\le k<N} \frac{T(k+1)}{(N+1)} \times \sum_{0\le i\le k} P(i,k) \times l^i(1-l^{i+1})^{N-k-1} \end{aligned}$$

$$\begin{aligned} E[\#\ messages]_{\Delta=0} &= \sum_{1\le i\le N} i \times Pr[exactly\ i\ messages\ sent] \\ &= \sum_{1\le i\le N} i \times P(i,N) \end{aligned}$$

3.7.2 Decaying Exponential Distribution

For this case, $p(t) = e^{-t/\alpha}/\alpha$, $P(t) = 1 - e^{-t/\alpha}$. Let $c = (1-l)$ and $Q(t) = cP(t)$. Then $q(t) = cp(t)$ and $q(t) = dQ/dt$.

$$\begin{aligned} E[t_{min_e}] &= \frac{-Tl^N + \int_0^\infty (1-Q(t))^N\,dt}{(1-l^N)} \\ &= \frac{-Tl^N + \int_0^\infty (1-cP(t))^N\,dt}{(1-l^N)} \\ &= \frac{\alpha}{N}\left[\frac{-Tl^N N}{(1-l^N)\alpha} + \frac{(1-l)^N}{(1-l^N)}\right] \end{aligned}$$

$$\begin{aligned} E[t_{min_k}] &= \int_0^\infty Ntp(t)\binom{N-1}{k}P(t)^k(1-P(t))^{N-k-1}\,dt \\ &= \int_0^\infty \frac{Nt}{\alpha}\binom{N-1}{k}\left(1-e^{-t/\alpha}\right)^k e^{-t(N-k)/\alpha}\,dt \end{aligned}$$

While we are unable to produce a closed form solution to the integral for $E[t_{min_k}]$, in Section 3.8 we numerically integrate the formula and display results for when N is an integer.

$$\begin{aligned} E[t_{max}]_{\Delta=0} &= \sum_{0\le k<N} t_{min_k} \times Pr[t_{max} = t_{min_k}]_{\Delta=0} \\ &= \sum_{0\le k<N} t_{min_k} \times \sum_{0\le i\le k} Pr[exactly\ i\ messages\ sent] \times l^i(1-l^{i+1})^{N-k-1} \\ &= \sum_{0\le k<N} t_{min_k} \times \sum_{0\le i\le k} P(i,k) \times l^i(1-l^{i+1})^{N-k-1} \end{aligned}$$

Metric	*Description*
t_{min}	earliest time sent
t_{minr}	earliest time sent and received, t_{min_e}
t_{max}	latest time sent
$avg\ t_{min}$	average earliest time received
$avg\ t_{max}$	average latest time received

Table 3.2: **Time Metrics Simulated.**

$$E[\#\ messages]_{\Delta=0} = \sum_{1 \le i \le N} i \times Pr[exactly\ i\ messages\ sent]$$
$$= \sum_{1 \le i \le N} i \times P(i, N)$$

3.8 Analysis and Simulation

In our initial simulations, N, T and Δ were fixed. Each simulation was run 1000 times when $N \le 100$, 100 times when $100 < N \le 500$, and 20 times when $500 < N \le 1000$.[2] Each node chose a delay time, t_i, based on the Unix `srandom()` function that had been seeded with the simulation start time. The simulations specifically explored the behavior of the Suppression algorithm with increasing packet loss probabilities, ranging from 0 to 1 in increments of 0.1.

Below, we compare performance along several axes. First we study the impact of loss on the various performance metrics; time and messaging overhead. We follow this with an analysis of the differences between correlated and uncorrelated loss. In addition, we contrast the uniform versus exponential distributions, expanding upon the results in Chapter 2.

3.8.1 Time Metrics

In Table 3.2 we summarize the values we tracked in our simulations for the Suppression with Loss algorithm. We were interested in definitive values for the earliest (t_{min}) and latest (t_{max}) messages sent, as well as the earliest message both sent and received ($t_{minr} = t_{min_e}$). In addition, we tracked the average values ($avg\ t_{min}$ and $avg\ t_{max}$) across all processes, since uncorrelated loss causes different processes to receive different messages.

Minimum Time Elapsed. Uncorrelated loss leads to smaller simulated t_{minr} (t_{min_e} in our analysis) than correlated loss for small N (Figure 3.4). However, the values converge with large N, and the point of convergence shifts upward (in N) as l increases (Figure 3.5). Simulations show that the uncorrelated t_{minr} values are identical to t_{min} in the lossless case, up to a large percentage of loss

[2] For the simulation variable t_{minr}, which identifies the earliest time of a message both sent and received, and which is discussed in the next section, the normalization factor was the number of iterations where not all messages were lost.

(Figure 3.6). This suggests that there is always one process within the group of participants that receives the smallest time selected, other than the sender of the message. However, once loss levels are extremely high, the group size N must be higher to counteract the effects of message loss.

In the correlated loss case, the analytic results for t_{min_e} match the simulation results for t_{minr}, which is evident in Figures 3.4, 3.5, and 3.6. Thus t_{min_e} serves as a good upper bound on the behavior of t_{minr}.

Nonetheless, the averaged values ($avg\ t_{min}$) are very close for uncorrelated and correlated loss. For uncorrelated loss, $avg\ t_{min}$ slightly underestimates t_{min_e} for small N, and overestimates it for large N.

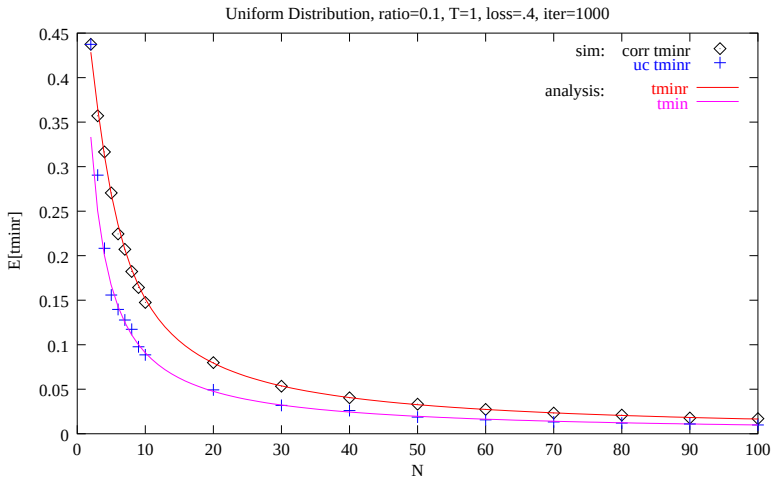

Figure 3.4: $E[t_{minr}]$ **vs.** N **(**$l = .4$**): Correlated vs. Uncorrelated.**

Maximum Time Elapsed. As the number of processes N increases, the value t_{max} becomes asymptotic; above a certain number of processes the same $E[t_{max}]$ (or $E[avg\ t_{max}]$) results (Figure 3.7). Not surprisingly, as packet loss l increases, the point at which the curves flatten out increases, i.e., it requires the participation of many more processes before the asymptote is reached. In effect, greater numbers of N are required to offset the impact of greater l.

For the parameters selected, the asymptotic value begins at $\Delta = .1$ for all t_{max} values (average, definitive, correlated and uncorrelated), and increases as l increases. As expected, when there is little loss, the message with the minimum time will reach and suppress most other processes, except for those selecting a t_i within Δ of the minimum time. As more messages are dropped, fewer processes receive the message containing the minimum time, thus resulting in additional messages generated, which results in the last message being sent further out in time.

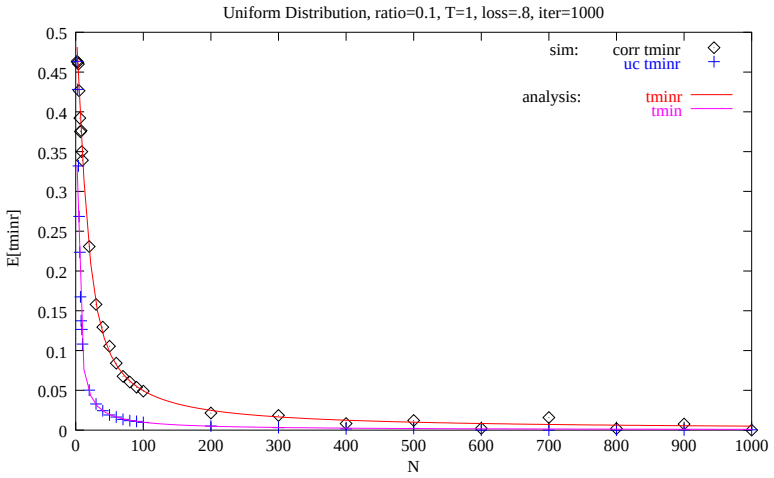

Figure 3.5: $E[t_{minr}]$ **vs.** N $(l = .8)$**: Convergence.**

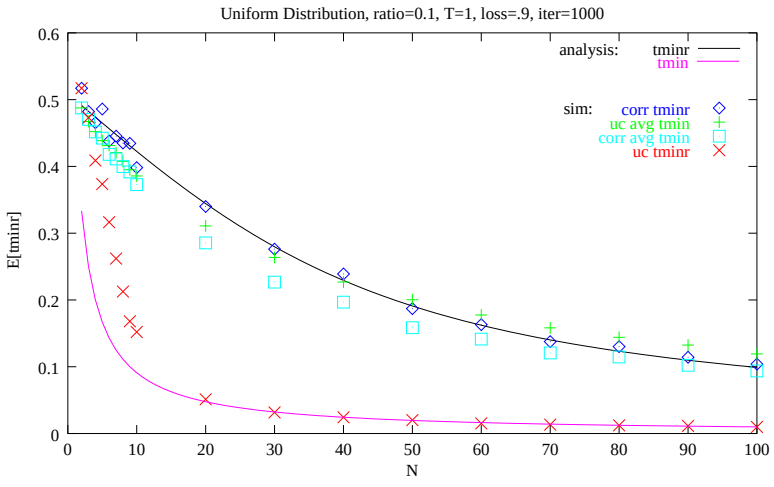

Figure 3.6: $E[t_{minr}]$ **vs.** N $(l = .9)$**: Large Loss.**

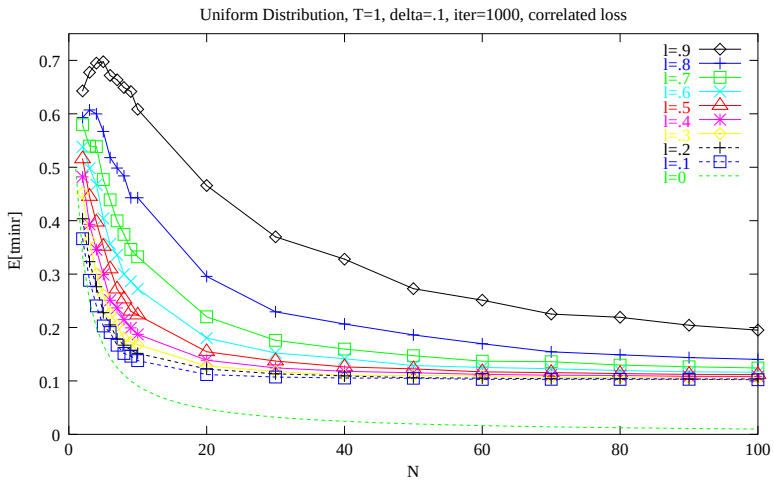

Figure 3.7: $E[t_{minr}]$ **vs.** N**: Varying** l**.**

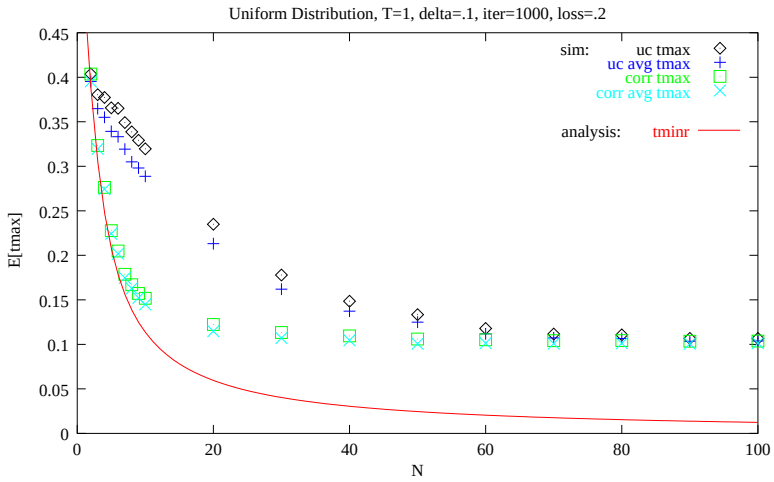

Figure 3.8: $E[avg\ t_{max}]$ **and** $E[tminr]$**: Small Loss** $(l = .2)$**.**

As packet loss increases, values of t_{max} and $avg\ t_{max}$ diverge, with average values being lower than definitive values and correlated loss values being lower than uncorrelated loss values (Figure 3.8). At loss rates of $l \geq .5$ (Figure 3.9), the average and definitive values for uncorrelated loss begin to diverge significantly.

In the uncorrelated case, the shape of the t_{max} curve changes substantially when the message loss l is high. The value of t_{max} increases at first, plateaus, and eventually decreases again to reach an asymptotic value. To explain this phenomenon we consider the probability that there exists a *straggler* process. We define a straggler as a process that loses all messages sent except its own. Consider the scenario where all processes send messages. Let l equal the probability that a message is lost and $N-1$ equal the number of other processes (besides self). The probability that a process

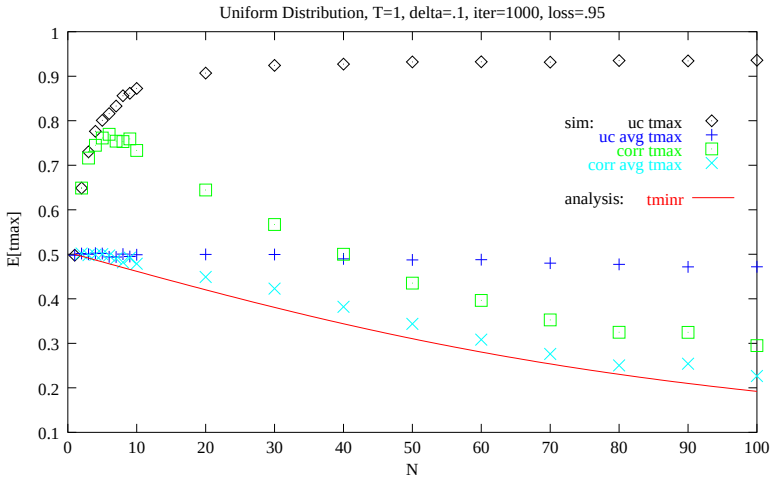

Figure 3.9: $E[avg\ t_{max}]$ **and** $E[tminr]$**: Large Loss** ($l = .95$)**.**

is a straggler:

$$\begin{aligned} P_s &= Pr[a\ process\ is\ a\ straggler] \\ &= l^{N-1} \end{aligned}$$

The probability there exists a straggler in a group of N processes is the same as 1 minus the probability that no stragglers exist in the group.

$$\begin{aligned} P_{s_N} &= Pr[there\ exists\ at\ least\ one\ straggler\ in\ a\ group\ of\ N\ processes] \\ &= 1 - (1 - P_s)^N \\ &= 1 - (1 - l^{N-1})^N \end{aligned}$$

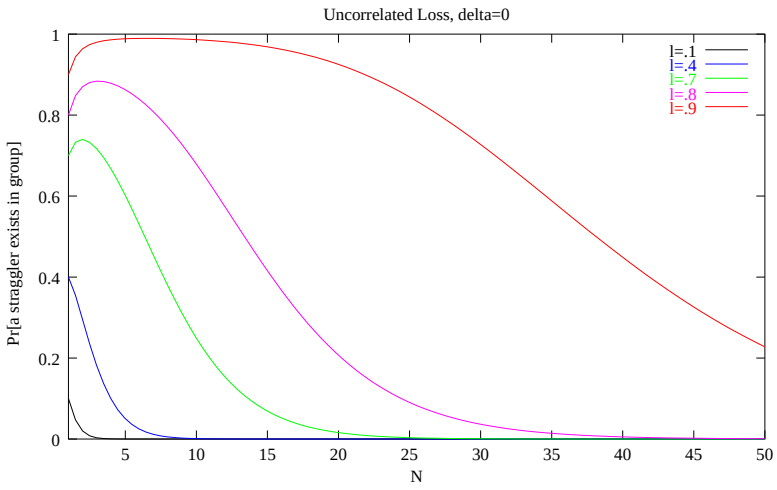

Figure 3.10: **Probability of a Straggler in a Group of Size N.**

These calculations show that small group sizes are prone to stragglers, as are groups with high loss rates (Figure 3.10). In the Suppression algorithm even smaller numbers of processes actually generate messages, so this phenomenon is even more pronounced.

Note also that, as N grows, the value of the largest t_{min_k} increases. For example, when time selection is from a uniform distribution, the largest t_{min_k} approaches the bound on the timer interval T. The impact of a single straggler on t_{max} is potentially greater with larger N, but has a lower probability.

For uncorrelated loss, as $l \to 1$, $avg\ t_{max}$ approaches the mean of the distribution (Figure 3.9). This occurs because, when all N messages are dropped in the network, each node does not receive any of the $N-1$ announcements of its peers. In that case, a node's locally selected time becomes the suppression time, as well as the minimum time received. Basically, each node reverts to using its own value for suppression. Each process i awakening at t_i sends its message. Although none of the messages are received, each individual node believes it has suppressed the others. In fact, there is no way for a process to distinguish between when it has suppressed all its peers and when all the peers' messages have been lost.

For correlated loss, as $l \to 1$, $avg\ t_{max}$ approaches the analytic results for t_{min_e} (Figure 3.9).

Uncorrelated loss consistently produces a larger than or equal t_{max} than the value produced under correlated loss. With small l the uncorrelated and correlated loss graphs converge as N increases (Figure 3.8). With increases in l, convergence no longer occurs not even at larger N (Figure 3.9). However, the individual plots are still asymptotic.

Finally, we note that the simulations match the analysis presented earlier (Figure 3.11).

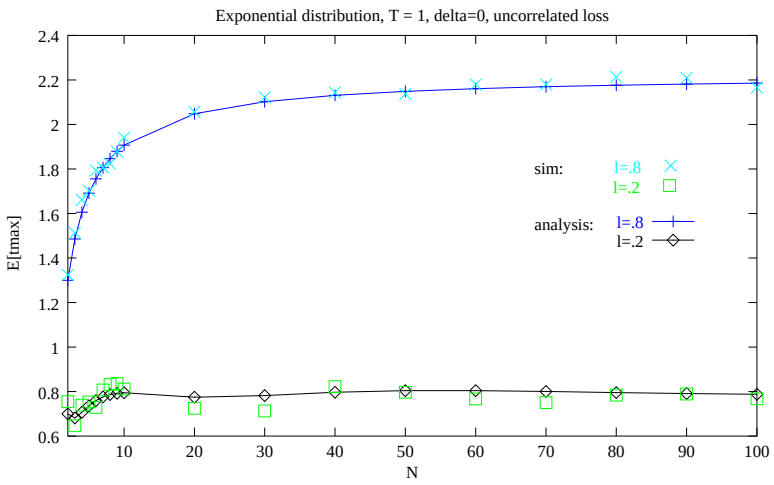

Figure 3.11: $E[t_{max}]$ **vs.** N**: Simulation vs. Analysis**

Averages. We observe from the graphs that, regardless of loss model, $avg\ t_{min} > t_{min}$ and $avg\ t_{max} < t_{max}$. When packet loss occurs, some processes will lose the message containing the definitive minimum time sent, thus the average becomes higher than t_{min}, because additional suppression messages will be generated which have later times. Likewise, when some processes lose the message containing the definitive maximum time sent, the average becomes lower than t_{max}.

3.8.2 Messaging Overhead Metrics

In Table 3.3, we summarize the metrics tracked in our simulations for the Suppression with loss algorithm. The parameter *num* represents the number of messages sent, *extra* indicates the number of extra messages generated, and *required*, the number of messages necessary to completely suppress all participating processes. The *avg num* parameter tracks the average number of messages received by the processes, as each process may receive different numbers of messages in the case of uncorrelated loss. The results reported below apply equally well to both uniform and exponential random timers.

Metric	*Description*
num	number messages sent
extra	number extra messages sent
required	number required messages sent
avg num	average number messages received

Table 3.3: **Messaging Overhead Metrics Simulated.**

Messages Generated. Regardless of message loss level, $E[num]_{corr} < E[num]_{uc}$, and becomes more apparent with larger N and with greater loss (Figures 3.12 and 3.13). In addition, the average number of messages received is approximately the same for both the uncorrelated and correlated

cases. $E[num] > E[avg\ num] > E[required]$ holds across all l. As Δ increases in size, $E[extra]$ becomes a larger component of $E[num]$.

In Figure 3.14, we show that simulations validate the analysis of $E[\#\ messages]_{\Delta=0}$.

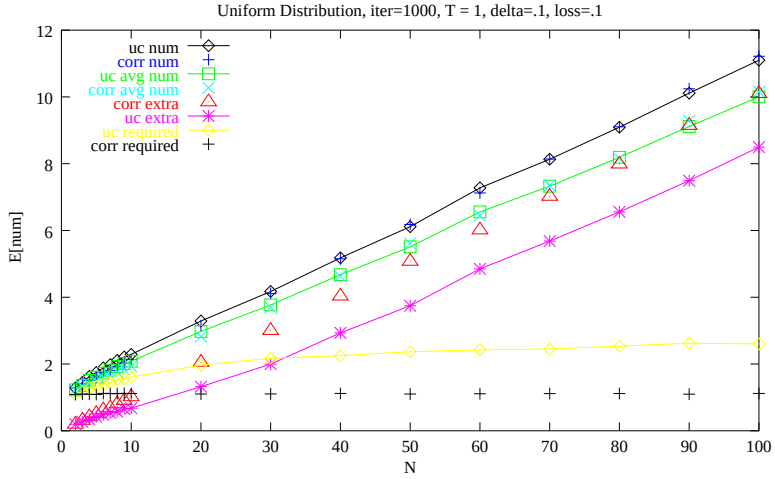

Figure 3.12: $E[num]$ **vs.** N $(l = .1)$**: Correlated vs. Uncorrelated.**

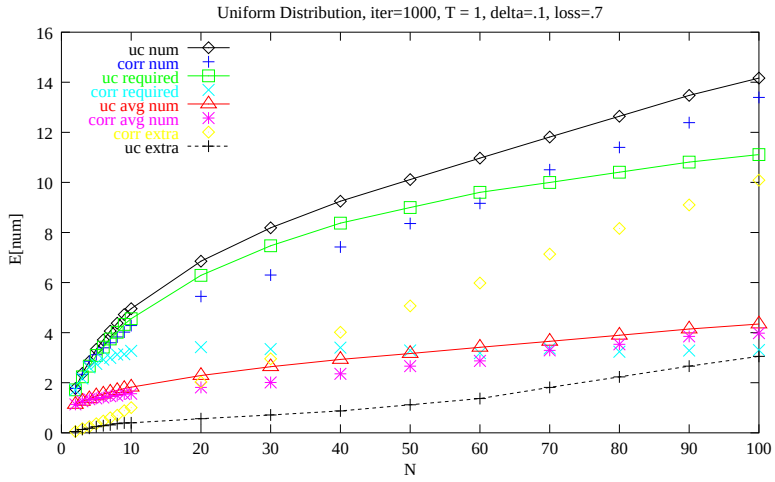

Figure 3.13: $E[num]$ **vs.** N $(l = .7)$**: Correlated vs. Uncorrelated.**

Extra and Required Messages. In Figures 3.12 and 3.13, we also observe that $E[required]_{uc} > E[required]_{corr}$, and because of the inverse relationship between required messages and extra messages (due to the definition of required messages), $E[extra]_{corr} > E[extra]_{uc}$. As $l \to 1$, the ratio of required messages to messages sent rapidly approaches 1, meaning many more messages are needed for Suppression to work properly. More importantly, $E[required]$ becomes asymptotic as N increases, for all l. The implication is that beyond some N, the same number of messages are

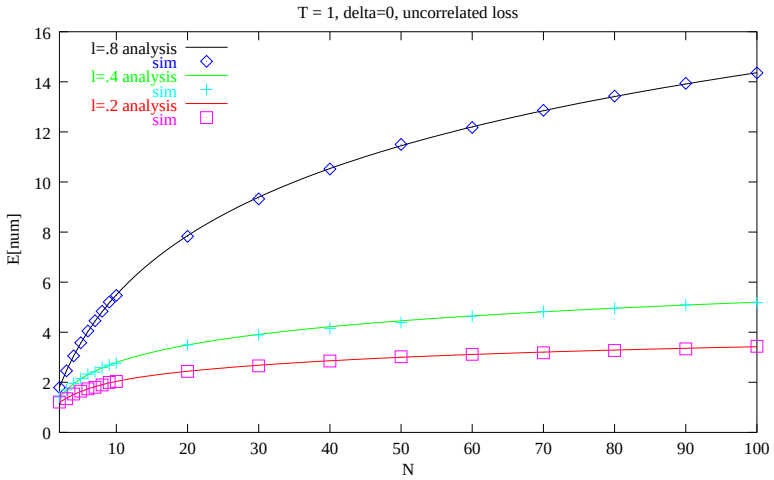

Figure 3.14: $E[\#\ messages]_{\Delta=0}$ **vs.** N**: Simulation vs. Analysis.**

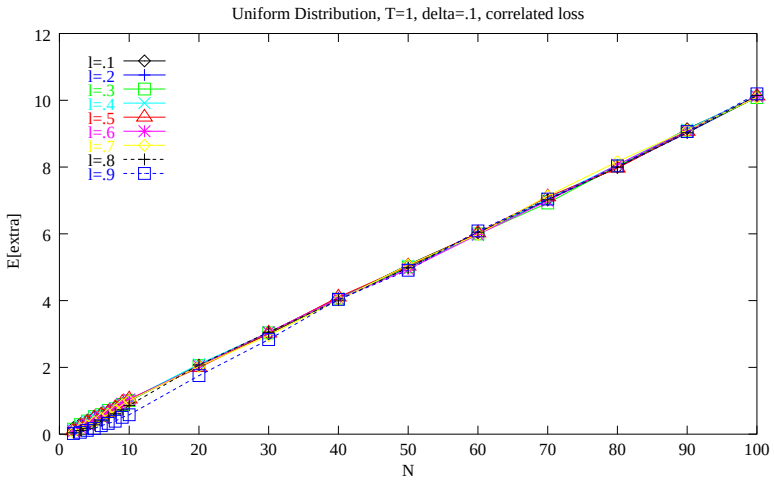

Figure 3.15: $E[extra]$ **vs.** N**: Correlated Loss (Varying** l**).**

necessary for Suppression to operate correctly.

In addition, $E[extra]_{corr}$ is nearly constant across loss levels (Figure 3.15). This phenomenon is due to the fact that, in the correlated case, there is only one message that suppresses all the processes; if it reaches one remote process, it reaches all of them. Therefore, the number of extra messages will be the number of messages arriving within Δ of the message that suppresses the sender. This number is a function of the density of the t_{min_k} values and the ratio of Δ to T; for example, with a uniform distribution where the expected values of the t_i are equally spaced, one would expect that a Δ/T ratio of .1 would lead to $N/10$ extra message arrivals, which corresponds with simulation results.

3.8.3 Loss Models

To summarize our observations relating to the loss model, we note that uncorrelated loss produces lower $E[t_{min_e}]$, as well as higher $E[t_{max}]$. This is due to the fact that probabilistically it is likely that at least one process other than the sender receives the earliest message generated, and probabilistically when N is large enough there will exist an outlier process that does not receive multiple messages sent. Thus, probabilistically uncorrelated loss will produce longer delays before the last message is sent.

Moreover, correlated loss leads to an optimal message ordering. Because all messages are either lost or received, each message sent will be in T_{min} order. The first message will be sent at t_{min_0}. If that is unsuccessful, it will be followed by one sent at t_{min_1}, followed by at t_{min_2} and so forth. Uncorrelated loss doesn't have that property. The earliest message, while lost by some processes, may be received by others. Therefore, t_{min_0} would be followed by t_{min_i}, where i is at best equal to 1, but will be the minimum of the remaining unsuppressed processes.

As a result, uncorrelated loss leads to higher $E[messages]$ and $E[required]$, whereas correlated loss leads to higher $E[extra]$.

3.8.4 Distributions

As in the lossless case, the uniform distribution performs better than an exponential distribution with regard to timing metrics, but performs worse with regard to messaging overhead metrics. These findings typically are more pronounced the larger the loss.

This makes more sense in light of what we know about the distribution of $E[t_{min_k}]$. As k increases, the expected values for t_{min_k} remain equally spaced from each other in the uniform distribution, while they are increasingly further apart for the exponential distribution (Figures 3.16 and 3.17).

For a metric such as $E[t_{max}]$, the expected time of the last Suppression message generated, this can result in substantial performance differences between the uniform and exponential distribution,

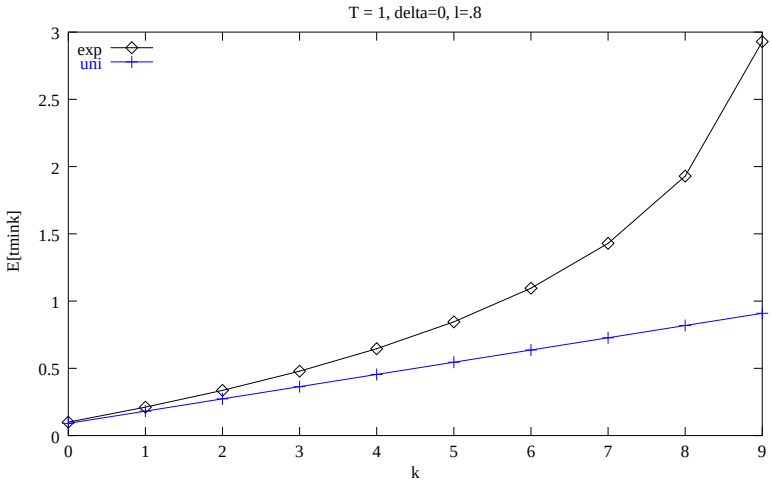

Figure 3.16: $E[t_{mink_k}]$ **vs.** k**: Uniform vs. Exponential (Small** N**).**

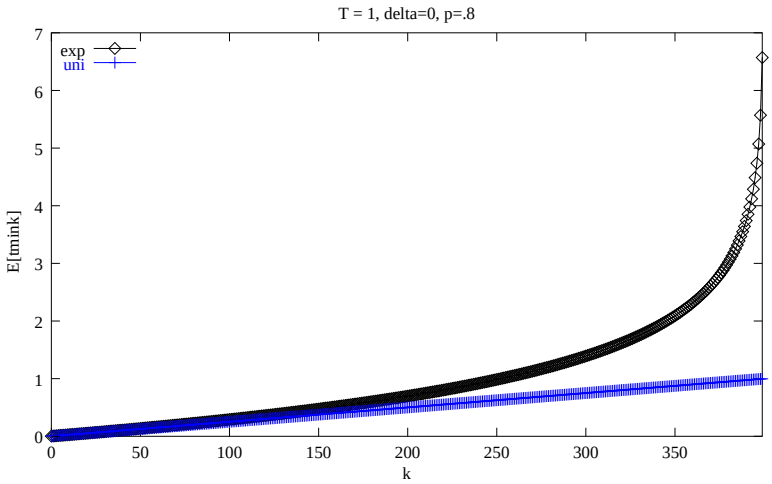

Figure 3.17: $E[t_{mink_k}]$ **vs.** k**: Uniform vs. Exponential (Large** N**).**

depending on the loss model. Thus, if t_{max} is an important delimiter for an algorithm, then the choice of an exponential distribution should be reconsidered. Although the differences are most notable with $N < 40$ in the correlated case (Figure 3.18), in the uncorrelated case the differences are noticeable up to group sizes $N = 100$ for moderate loss levels ($l = .4$), and even higher as p increases (Figure 3.19).

An interesting phenomenon is that the average number of messages received decreases as loss rate increases. This holds for both distributions. Although we only display a graph showing correlated loss, this also holds for uncorrelated loss (Figure 3.20). In the uncorrelated case, each process receives one message that constitutes the earliest Suppression message plus it receives any other messages generated as a function of the local process having a t_i within Δ of t_{min_e}. As p increases, fewer

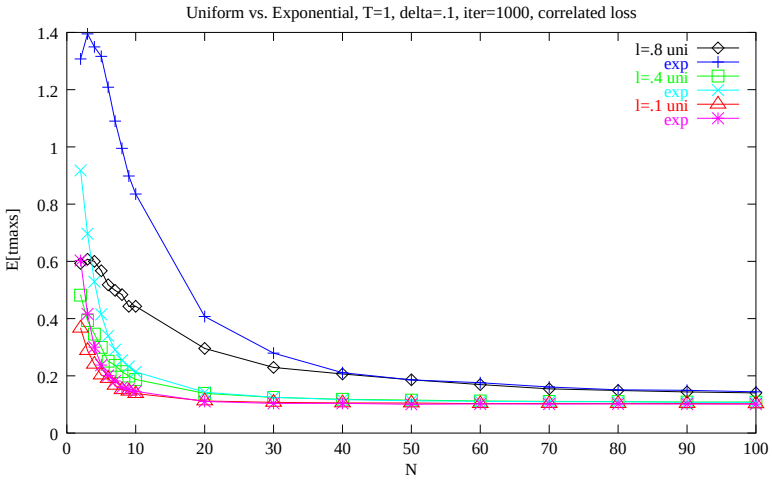

Figure 3.18: $E[t_{max}]$ **vs.** N**: Correlated Loss.**

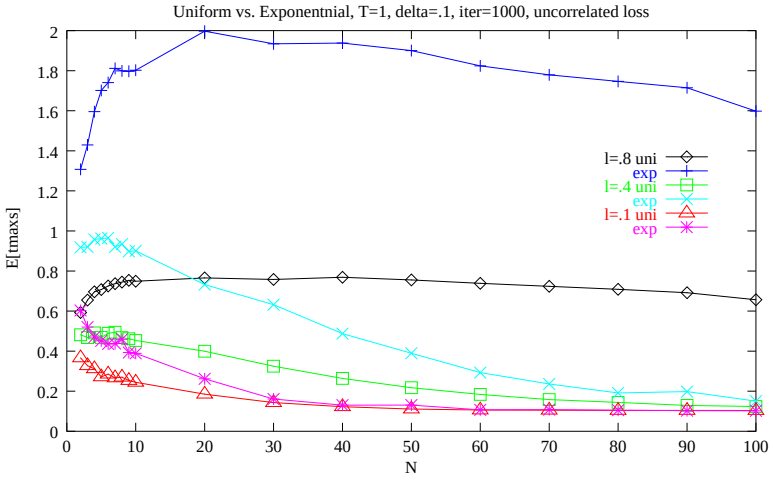

Figure 3.19: $E[t_{max}]$ **vs.** N**: Uncorrelated Loss.**

of the extra messages successfully get through and the overall average number of messages received drops. As $p \to 1$, the average number of messages received should approach 1, as fewer and fewer extra messages are delivered successfully. In the correlated case, fewer extra messages are delivered, but proportionally fewer of the Suppression messages are delivered as well, since with uncorrelated loss each process may receive multiple Suppression messages before the algorithm completes.

Similarly, as $p \to 1$, $E[\#\ extra]$ decreases, regardless of distribution. As stated previously, uniform distributions, which exhibit better response times, produce more extra messages than the exponential distribution, particularly with uncorrelated loss.

For a particular level of message loss, the number of required messages is constant across distributions. In other words, the uniform and exponential distributions lead to identical $E[required]$ (Figure 3.21). This is due to the fact that required messages are defined as the number of messages

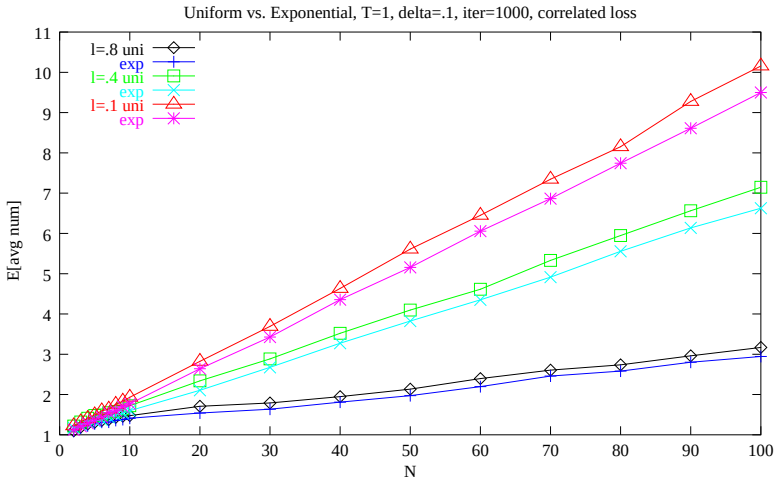

Figure 3.20: $E[avg\ num]$ **vs.** N**: Correlated Loss.**

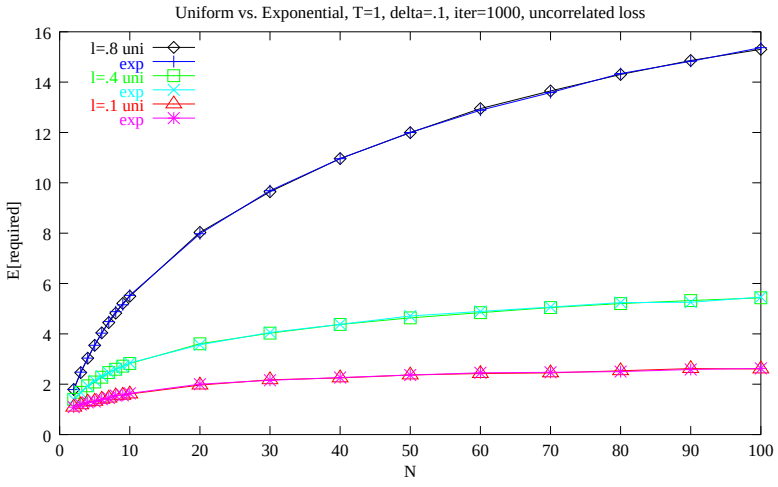

Figure 3.21: $E[required]$ **vs.** N**: Uncorrelated Loss.**

generated by the algorithm when there is zero delay present; in Section 3.4 we show that this metric is purely based on $P(i, N)$, a function only impacted by message loss. Therefore, the distribution for the Suppression interval is rendered immaterial.

3.9 Related Work

Observation of the MBone revealed that there were three general classes of loss that occur [37]; (1) 25% of receivers suffered no loss throughout the day, (2) of those experiencing loss, the median loss rate was between 5-10% for the majority of the day, and (3) a full 25% of the receivers experienced loss rates greater than 15% all afternoon. Essentially, the loss distribution was very long tailed, with large amounts of relatively low loss rates. A full 80% of all nodes reported loss rates less than 20%. Yet, these results suggest that simulations with only single packet losses easily underestimate realistic loss conditions [27] [50] [47].

Handley's results [37] corroborate findings in an earlier study conducted by Kurose et al. [59], which found that loss on backbone links was small, as compared to the average loss observed by a receiver. That is, much of the loss occurred at the edges of the network close to senders or receivers. The result is that the pattern of loss is closer to the extremes: fully correlated or fully uncorrelated.

Both studies confirm the existence of loss correlation. However, Handley [37] also found that there always exists a small number of receivers suffering very high uncorrelated loss.

Although there have been several studies that analyze the performance of Suppression with loss (in the context of reliable multicast protocols, in the form of two back-to-back Suppression phases), loss is modeled in fairly simplified terms [27] [50] [45].

While Floyd et al. offer a detailed examination of the effects of topology on Suppression, they only study the impact of a single loss of original data, not the loss of repairs and requests [27]. Even though this is a good first step toward understanding the impact of loss on Suppression, it falls short of realistic conditions. Raman et al. [50] extend the work in [27], focusing on understanding the impact of large group size on the number of duplicate NACK messages. Again, the loss model only studies the effects of single packet losses on the algorithm.

Nonnenmacher and Biersack [45] simulate slightly more complicated loss scenarios, in terms of where the losses occurs (i.e., between the sender and receivers or between the receivers themselves during a repair), and the full range of loss levels; however, they do not provide accompanying analysis.

Birman et al. analyze a bimodal reliable multicast protocol under probabilistic failures [10] [39], but only examine independent identically distributed losses.

Whereas several previous studies concentrate on the impact of single losses and/or losses close to the source [27] [50] [45], this thesis studies multiple simultaneous losses (that occur anywhere in the

system), it looks at the full spectrum of loss rates, and it considers uncorrelated loss and correlated loss.

Although there are other studies, such as the ECSRM [33] and SHARQFEC [41] protocols, that study the scalability of Suppression and simulate the effects of multiple losses on it, the research does not focus on the analytic derivation of performance metrics. Nonnenmacher's work derives metrics analytically for Suppression in the lossless case [46], however no analysis of metrics is presented for either the lossy case or the combined loss and delay case.

3.10 Summary of Results

In this chapter we analytically derived Suppression metrics under the combined conditions of loss and delay, or simply in terms of loss with zero-delay when the analysis proved intractable. We re-examined the metrics introduced in Chapter 2 and introduced several new metrics to characterize the performance of Suppression.

We revisited the meaning of the metric $E[t_{min}]$, offering a more realistic definition of the minimum response time of the algorithm, $E[t_{min_e}]$, the expected time of the earliest message sent but not completely dropped in the network. In other words, the earliest hope of Suppression by another process. To complement this, we presented a bound for $E[t_{max}]$, the expected time after which all processes are considered suppressed, i.e., the completion time of the algorithm. We defined the vector $\vec{T}_{min}$, an ordering of the Suppression wake-up times selected, and calculated $E[t_{min_k}]$, for $0 \leq k < N$, for each probability density function. The comparison explained why the uniform distribution outperforms the exponential distribution for time-related metrics, and vice versa for overhead-related metrics.

On closer examination of $E[\#\ extra]$ we found that it is just one component of $E[\#\ messages]$, the total number of messages generated by the algorithm. Although we were unable to derive a closed form expression to describe $E[\#\ messages]$ analytically, we were able to observe its behavior in simulation and offered a recurrence relation to explain $E[\#\ messages]_{\Delta=0}$, the number of messages generated in the zero-delay case. We postulated that $E[\#\ messages]$ is at least as great $E[\#\ messages]_{\Delta=0}$.

$E[\#\ extra]$ was redefined more broadly as the overhead of the algorithm, whereas $E[\#\ required]$ was established as the expected number of messages required to fully suppress a group of size N. Another way to think about $E[\#\ required]$ is as the number of messages generated when there exists no transmission delay. The implication is that the number of required messages remains constant across all Δ. More importantly, the number of required messages is the same across distributions, as it only has a loss component and no delay component. Finally, $E[\#\ required]$ is asymptotic. That is, beyond a certain N, the same number of messages are necessary for Suppression to work

properly. In the next section, we discuss the important implication of this result; perhaps a redesign of the Suppression algorithm would optimize performance further.

Our analysis of the metrics was validated through extensive simulations, which employed a more sophisticated and more realistic loss model, compared to all known studies of Suppression analytically and all of which limit the number of dropped messages, the placement of the loss in the topology, or the level of lossage. For each metric, we presented bounds for worst-case performance, after a thorough examination of both the correlated and uncorrelated loss scenarios. Finally, our simulation and analytic results matched.

3.11 Future Work

As stated in Chapter 2, there are refinements that would be beneficial to make to the model. In particular, we would like to re-evaluate these metrics under conditions of variable delay.

Presently, we have expressions for $E[\#\ messages]_{l=0}$ and $E[\#\ messages]_{\Delta=0}$, which are complementary. However in the future, we would like to derive an expression for $E[\#\ messages]$ with both delay and loss present in the system. This in turn would allow an analytic solution for $E[\#\ extra]$. In addition, it would be helpful to have a closed form solution for $E[t_{min_k}]$ in the exponential case when making the assumption that N is an integer. Similarly, a general expression is sought for $E[t_{max}]$ in the presence of both delay and loss.

An interesting byproduct of knowing $E[\#\ required]$, given the other system parameters p, T, N, and Δ, is that we can calibrate a *believability* factor. In other words, a node must decide whether or not to believe that the correct course of action is to suppress itself when it receives a suppression message from another node. Or should it send its message anyway? For instance, if $E[\#\ required]$, the number of messages required to suppress all N nodes is three, and a receiver has only received two messages before it awakens and must choose whether to suppress or to send its own message, then perhaps it should send its message anyway.

To examine this thread of reasoning, we could create a new parameterized Suppression algorithm that sends a certain number of messages at wake-up, as shown in Program 3.1. The number of messages to send at wake-up would be based on a function $g(m)$ where m is the number of messages already received. While this approach may be quite effective when Δ/T is small, it may be much less so as Δ becomes a substantial part of the timer interval, T.

Alternatively, to minimize $E[t_{max}]$, the node that awakens earliest could issue $E[\#\ required]$ messages immediately. Another scheme could send the $E[\#\ required]$ messages from different senders by basing the to-send-or-not-to-send decision on a probabilistic estimate. For instance, if the time t_i selected by node i is smaller than $E[t_{min_r}]$, where $r = \#\ required$, then issue a message immedi-

```
SUPPRESSION-PARAMETERIZED (d, T, N, Δ, p))
1    t = random(d, T)
2    sleep(t)
3    m = num_messages_received()
4    r = required_messages (d, T, N, Δ, p))
5    if (m < r) then
6       send_message(g(m))
```

Program 3.1: **Parameterized Suppression Algorithm.**

ately, thus sending the messages from different topological regions. Because $E[\#\ required]$ becomes constant beyond a particular group size, these algorithms would be robust to large fluctuations in group membership. These approaches of course may lead to additional extra messages (or other drawbacks). Future investigation should explore if there are benefits across the different loss models and random timer distributions.