PhysComp96: `selforg`
Full paper
Draft, November 23, 1996
`smith: jrs@media.mit.edu`

Self Organizing Device Networks

Joshua R. Smith
Physics and Media Group
MIT Media Lab

In order for simple, initially identical computational units to acquire unique identities, two ingredients are needed: a physical symmetry breaking mechanism and a distributed algorithm that makes use of this mechanism. After briefly considering non-thermodynamic sources of symmetry breaking, I return to thermodynamic symmetry breaking, and present three algorithms that make use of this source. I analyze the expected resource requirements (time, random bits, space, communication) of the algorithms, and note a trade off among these measures of algorithm complexity. I close by speculating about a framework for formalizing these trade-offs and determining the theoretical optimum performance of symmetry breaking protocols.

1 Introduction and Motivation

In the next several years, computational, sensory, and communication capabilities will diffuse out of their present home in beige boxes on desktops and into everyday objects such as furniture, clothing, and even into things as insubstantial as balloons.[6] As the cost of electronics continues to drop, the number of networked, activated devices will grow rapidly. Soon you will probably be able to telnet to your toaster (if you can't already). To communicate over a multiaccess channel, each one of these billions of devices will need a unique identifier. Present communication protocols such as Ethernet specify that the manufacturers must communicate with one another, through the IEEE, in order to avoid assigning the same ID twice.[9] This paper explores methods by which the *devices* could coordinate with one another to manage ID assignment dynamically and automatically.

In particular, this paper is an investigation of distributed protocols that utilize physical sources of symmetry breaking to enable a network of initially identical units to acquire the unique identities required for point-to-point communication over a shared resource such as a bus or common RF band (multiaccess channel). I present several protocols, compare their resource use (time, random bits, space, communication), and note a trade-off among these measures of algorithm complexity. I close by speculating about a framework for formalizing these trade-offs and determining the theoretical optimum performance of symmetry breaking protocols.

The goal, once again, is that each initially identical unit have a unique identifier by the end of the protocol. Unique identifiers, or at least sources of symmetry breaking, are necessary for point-to-point communication over a shared channel: if two units have duplicate IDs, both will respond to the same set of messages, and when both try to respond simultanesouly, there is no way for them to break the deadlock, since the two units begin in the same state and proceed deterministically, in particular pausing and retransmitting at the same time. The "random exponential backoff" used to recover from collisions in CSMA/CD networks (eg Ethernet) is impossible without either a source of thermodynamic randomness, or a unique ID such as the Ethernet MAC address that can seed a pseudo-random number generator.

Thus the protocols must use uncorrelated noise sources, together with communication, to move the system from a perfectly correlated state (all units with ID 0, say) to create a kind of perfect anti-correlation, in which each unit is in a unique state.

An obvious method for generating a unique ID is to hardcode it into the unit's ROM. In this solution, the symmetry is broken in advance of operation, in the course of the manufacturing process. Effectively this means that the manufacturing process for each unit is slightly different. Clearly the process of manufacturing a large number of unique units is more complex than that of manufacturing a large number of identical units, and this complexity is undesirable because it translates to cost. For an item as expensive as an Ethernet card (around \$50), the cost of the coordination activity required to ensure that each receives a unique ID is small compared to the cost of the unit. For a \$5 sensor unit, this coordination cost is relatively higher, and for active RFID (\$1) and passive tags (\$.01), the coordination cost begins to dominate the other manufacturing costs.

Dallas Semiconductor in fact sells a "silicon serial number" (for a around \$1 in large quantities) that is nothing but a 56-bit unique ID that can be read by a one-wire serial protocol. Each unit made is guaranteed to have a different ID than all others.[8] Effectively this manufacturer is mass producing broken symmetry, centralizing the manufacturing burden of producing unique items into a single production line.

Since the number of "smart" devices with unique IDs will be growing rapidly in the next several years as their cost drops, the idea of building the *capability* to break symmetry into each device, of moving the symmetry breaking step from the production line to the device itself, may become increasingly attractive. An interesting practical question is whether dynamic symmetry breaking schemes such as the ones we will present can be more efficient economically (ie cheaper) than the hard-wired, "mass produced broken symmetry" represented by Dallas' silicon serial number.

In addition to the possible cost savings, there is another practical motivation for understanding dynamic symmetry breaking protocols: they may enable anonymous communication between units, or between a unit and a public server. Since the identities are determined after the units have been deployed, there is no possibility that the manufacturer or other entity could maintain a list associating IDs with owners. However, in order to evaluate the protocols presented here from the perspective of preserving anonymity, we would have to consider the effect of adversaries who may not obey the protocol, which is beyond the scope of this paper.

1.1 Biological Examples

There are several well known biological examples of distributed symmetry breaking protocols. The "jamming avoidance" behavior of the South American weakly electric fish *Eigenmannia* is one such example.[3] Members of this species generate continuous sinusoidal electric fields of constant frequency for both sensing and communication purposes. For a fish to sense its environment effectively, it needs its own electrolocation frequency. When the sensing frequencies of two fish overlap enough to interfere with electrolocation, both fish can sense this collision, and the resulting beat frequency. The fish with the higher frequency raises its frequency still higher, and the lower frequency fish drops its frequency, resolving the conflict. Of course, this may create new frequency conflicts, which are then resolved by the same algorithm. In this way, the frequency spectrum used by the fish can be adaptively re-allocated when a new individual is introduced into a habitat.

Another natural example that has been discussed by Rabin in the computer science literature is the case of mites of genus *Myrmoyssus*, which feed on the eardrum of *Phaenidae* moths which in turn are fed on by bats that use sonar. Since the moths use their hearing to avoid the bats, it is in the mites' interest to eat one of the moths' eardrums at a time. What protocol do they use to chose an eardrum? Each mite emits the same concentration of a pheremone. The mites move toward the eardrum with higher pheremone concentration. Thus any initial asymmetry in the mite distribution is quickly magnified until all the mites are on one eardrum. In the computer science literature, this is referred to as the Choice Coordination problem. We can take these biological examples as existence proofs of distributed coordination schemes that make use of natural symmetry breaking.

Both of these examples share certain features with the problems we discuss: the initial part of the process involves symmetry breaking, and an increase in entropy, and the final part of the process involves a decrease in entropy, and results in some coordinated behavior (correlated in the case of the mites, and anticorrelated in the case of the fish). In the mite example, the symmetry breaking consists of deciding which of two identical ears to occupy first; the coordination portion occurs when all the mites move to this ear.

The physical origin of the symmetry breaking in the biological examples is not cut-and-dried. In this paper, we will be focusing on the use of thermal fluctation as the symmetry breaking mechanism, but this is not the only one. For example, if the units have different "world lines," that is, if they occupy different regions of spacetime, there are a variety of ways for the units to make measurements of the intervals separating them. The resulting internal record of the measurement can then be used to break symmetry and generate a unique ID. For example, if the units are stationary and at random locations, each can generate an electric field of the same amplitude. But because they are at random locations, generically (ie as long as the units are not in a special symmetrical arrangement) each will see a different field strength resulting from all the other units. This is also also the symmetry breaking mechanism employed by the mites (they are assumed to start at random positions).

Differences in "temporal location" can also be used to break symmetry: if each unit is powered up at a random time, then each can start a clock at power up. Since the clocks were powered up at random times, they will have different values.

For the remainder of this paper, we will focus on thermal fluctuations as the source of symmetry breaking. Thus we will assume that each unit has access to a source of random bits derived from thermodynamic fluctuations, such as Johnson noise in a resistor, or avalanche noise in a reverse-biased diode. In this paper, we propose and analyze several schemes that a set of identical units can use to turn idependent thermal noise into unique IDs.

2 The problem: Symmetry breaking and ID assignment

Like the examples mentioned above, the self-organizing ID assignment problem has two logically distinct subparts. The first part is to make all the units different in some way (to break symmetry), and the second part is to assign the unique IDs to the units, assuming that the symmetry has already been broken. (The two parts may not actually occur as two sequential phases, but they are logically separate.) In the first part, the entropy of the units' memory increases; in the second part it decreases. Self organizing processes are characterized by (local) lowering of entropy; but for this to occur in the context of our deterministic digital devices, a source of entropy, the thermal noise source, must be built in. The electric fish and the mites, by contrast, are "analog," and thus have ready access to fluctuations, inside their bodies, in the interactions between the bodies, and in the interactions of the bodies with the environment.

2.1 Related work on symmetry breaking problems

There are several problems that have been studied by computer scientists that have a symmetry breaking component to them. In all these problems, if symmetry is not broken in advance by pre-assigning a unique ID to each processor, then it must be broken in the course of the algorithm by use of independant sources of random bits (or by other physical symmetry breaking mechanisms, though this is never considered in the computer science literature).

The first well known symmetry breaking problem is the Dining Philosophers problem, posed in 1965 by Dijkstra[2]. In this problem, processes with overlapping resource requirements must be scheduled. The symmetry breaking problem arises when two processors try simultaneously to access the same resource: to break or avoid deadlock, one must yield. Dijkstra proposed a randomized algorithm in which each procesor waits for a random interval before trying to use the resource again.

Around 1971 a similar randomized protocol was used to avoid deadlocks after collisions in Alohanet, an early packet radio communication network, and a substantially improved, adaptive version of this scheme was developed in 1973 for Ethernet, and is still used in present-day CSMA/CD networks. In a CSMA/CD network, the size of the post-collision random wait window is doubled after each repeated collision, so that in heavy traffic, the processors wait longer before trying to retransmit. This random exponential backoff or binary exponential backoff leads to much more efficient channel use.[9]

In a 1982 paper, Rabin explored randomized algorithms for the Choice Coordination Problem, in which n processors must agree on a common choice of one out of k outcomes.[7] Awerbuch *et al.* present randomized algorithms for the maximal independant set and Dining Philosophers problem for an asynchronous distributed network of processors with arbitrary network topology.[1] These problems are sometimes referred to in the computer science literature as symmetry breaking problems because in order to solve them, symmetry must be broken, either by providing unique IDs or independant sources of random bits. Almost all the computer science work on symmetry breaking problems differs from ours in that it considers specific computational problems that have a symmetry breaking component. In this paper, we propose isolating the symmetry breaking problem, and solving it once and for all using an algorithm whose sole function is to assign unique identities. These identities can then be used to break symmetry if desired, and for additional non-computational purposes such as communication.

The one example in the computer science literature of using a symmetry breaking protocol to assign unique identities is a paper of Lipton and Park[4], in which they pose what they call the Processor Identity Problem—fundamentally the same problem discussed in this paper, but set in the context of a shared memory multiprocessing system. The protocol they describe is of no use in the multiaccess-channel scenario we are considering. Like most of the other work relating to symmetry breaking in computer science, the interactions between the processors (ie the communication model) is very different than ours. A very general framework for understanding symmetry breaking protocols would encompass all communication models, perhaps by allowing parameters such as shared memory requirements as additional measures of algorithm complexity.

2.2 Preliminaries

We will assume there are n units with T bits of storage each. Thus each unit may be in one of $m = 2^T$ states. The three schemes we will present each assume a different model of communication: the non-interacting algorithm assumes no communication, the sequential algorithm assumes a "wired-or" bus, and the parallel algorithm uses what I will call an Amplitude Modulation (AM) bus. On a wired-or bus, if any unit writes a one, all see a one. On the AM bus, all the units operating at a particular frequency can write a modulated carrier (modulated by zero or one, say), and can determine the amplitude of the sum of the (up to) n signals that were written on the bus at that frequency. So, the AM bus allows many simultaneous channels (different frequencies), and in each channel a sum of the input amplitudes is performed. Thus all the units in a communicating subgroup (cell) can learn the sum of the values they wrote onto the bus, typically the sum of their random bits.

In analyzing the proposed schemes, we will consider the scaling of three quantities as a function of n and p_f: the amount of storage, time needed to assign IDs, and the number of random bits used. We will also be mindful of the amount of communication used in each scheme, though it does not appear explicitly in any formulae.

2.3 Non-interacting solution and the "birthday" problem

Perhaps the most obvious solution is only to have a symmetry breaking phase: the random bits chosen by a unit in the symmetry breaking phase are its ID. This seems appealing intuitively, since the probability of a particular unit chosing a particular ID is very small, 2^{-T}. However, the probability of that some collision will occur is much higher. Recall the famous birthday paradox: given a set of more than 22 people, the probability that two or more of them will share a birthday is greater then one half. In particular, a simple combinatorial argument shows that the probability of failure p_f of the non-interacting protocol is

$$p_f = 1 - \frac{m!}{(m-n)!m^n} = 1 - \frac{2^T!}{(2^T-n)!2^{T^n}}$$

Following [5], we can bound this probability using

$$p_f \leq 1 - e^{-n(n-1)/2m} = 1 - e^{-n(n-1)/2^{T+1}}$$

Solving this inequality for T, we find

$$T \geq \log_2\left(\frac{-n^2+n}{\ln(1-p_f)}\right) - 1$$

Assuming that we are interested in a small probability of failure, we can make the approximation $\ln(1-p_f) \approx -p_f$, so $T \geq \log_2\left(\frac{n^2-n}{p_f}\right) - 1$. We now drop the additive constant, allowing ourselves one extra bit, and also drop the linear n term, which is negligible, to find the following useful expression for the number of bits needed in the symmetry breaking ID as a function of the number of units n and the desired probability of failure p_f:

$$T \geq \log_2 \frac{n^2}{p_f}$$

Since the algorithm has no other significant storage requirements, the space requirement is also given by this expression for T. For $n = 10^6$ and $p_f = 10^-15$, $T = 90$. Since, with high probability, the symmetry has been broken (each unit has a unique identifier), a compact set of working IDs in $0, \ldots, n-1$ can now be assigned.

The working IDs can be assigned by a binary search procedure. Assume for the moment that one unit can become "bus master" (below we will present a scheme by which this could happen). The bus master can broadcast a message asking whether there are any units with ID less than $\frac{1}{2}2^T$ and wait for a response. If there is a response, it cuts its pivot point in half and asks again. We do not need to assume that the bus master can detect communication collisions (multiple responses to its queries). We have chosen T large enough that the IDs are all distinct. Instead of stopping its search as soon as it has found a range of IDs containing just a single unit, the bus master can continue its binary search until it has discovered the precise "symmetry breaking ID" of each unit. Thus it can find an all the occupied IDs.

Finding a single occupied ID takes time $\log_2 m = \log_2 2^T = T$, and since this must occur n times, the total time for this procedure is nT. Each unit can then be assigned the working ID in the range 0 to $n-1$. The time needed to break symmetry is T and time nT is needed to assign the short IDs. Thus the total time needed is $L = (n+1)T$. Using the expression for T derived above, we can write the total time

$$L = (n+1)\log_2 \frac{n^2}{p_f}$$

The total number of random bits needed is $R = nT$; this can be written

$$R = n \log_2 \frac{n^2}{p_f}$$

2.4 Sequential symmetry breaking and ID assignment

For this scheme, the communication model is assumed to be a wired-or bus, a term that was defined in section 2.2. The n units engage in a tournement consisting of a series of n rounds. Each round ends with one winner, who takes the next unallocated ID, and then doesn't participate in later rounds. Thus the first tournement has n participants, and the last has just 1. In the terminology of distributed algorithms, this could be desribed as repeated leader election.

In each round, all the participating units chose a random bit, write it on to the shared bus, and then check the value of the shared bus (I will call this a trial). Because of the wired-or bus, if any unit writes a 1, all will read a 1. So if a unit writes 0 and reads 1, it knows that at least one other unit is sharing the bus and drops out of the round. Eventually, only one unit will remain; when it has read a long enough sequence of bits that are identical to what it wrote, it can be satisfied that it is alone, so it takes the next ID and broadcasts a message for the others to begin the next tournement.

The total space required by the algorithm is simply the number of bits needed to represent n IDs:

$$T = \log n$$

A rough analysis of the running time is as follows. There are two parts to the running time: $L = L_1 + L_2$. The first, L_1, is the expected time to produce a winner; it depends only on the number of units n. The second, L_2, is the time it takes the winner to realize that it has won. It depends only on the desired probability of failure p_f.

First we will estimate L_1. Round i, which has i participating units, consists of approximately $\log_2 i$ trials, since roughly half drop out of the round after each trial. Thus the total number of trials (total time) is approximated by

$$L_1 = \sum_{i=1}^{n} \log_2 i = \log_2 \prod_{i=1}^{n} i = \log_2 n! = \frac{\ln n!}{\ln 2} \leq \frac{n \ln n}{\ln 2}$$

To find L_2, we will evaluate our our "birthday failure" expression from the previous section with $n = 2$ to find the probability of 2 units chosing the same sequence of L_2 random bits. (We will neglect higher order failure modes, ie 3 or more units chosing exactly the same bit sequence.) The result is $L_2 = \log_2 \frac{1}{p_f}$. Thus

$$L = L_1 + L_2 \leq \frac{n \ln n}{\ln 2} + \log_2 \frac{1}{p_f}$$

The total number of random bits needed is

$$R = \sum_{i=1}^{n} i \log_2 i$$

For n sufficiently large, this sum can be approximated by an integral:

$$R \approx \int_{i=1}^{n} i \log_2 i di = \frac{1}{4}\left(n^2(\log_2 n - 1) + 1\right) \leq n^2 \log_2 2n$$

2.5 Parallel symmetry breaking and ID assignment

The goal is to split the n units into two equal groups after they have chosen T random bits, and then repeat the process recursively. Each of these iterations assigns one ID bit to each unit. If additional communication channels (eg frequencies) were available, then the units could divide into smaller and smaller communication cells, and the IDs could be assigned in parallel.

In this scheme, each unit first choses a random bit, which splits the group roughly in half; then the deviation from a perfect split is measured using the AM bus, and then further, corrective flips occur, with probabilities chosen so as to reduce the deviation. If a proposed random step ever increases the deviation, it is rejected, so the deviation converges monotonically to zero. For the corrective flip at time t, we chose the flipping probability p_t such that the expected value of the sum after the flip is $\frac{n}{2}$. We will denote the sum of the random variables ξ, and the expected value of the sum $\mu = \frac{n}{2}$. It will be convenient to define the "excess" $s = \xi - \mu$. Without loss of generality, we can always take s to be positive; s is negative indicates an excess of 0s, and in this case we can deterministically flip all the units, so that $s' = |s|$.

The correction step at time t is a biased random walk starting from ξ_{t-1} and moving in the direction of μ. The probability $p_t(\xi_t|\xi_{t-1})$ is a Gaussian with mean $\xi_{t-1} - p_t\xi_{t-1}$, because the initial value is ξ_{t-1}, and ξ_{t-1} is also the number of units eligible to flip from 1 to 0. At every time step we must chose p_t so that the expectation of $p_t(\xi_t|\xi_{t-1})$ is $\mu = \frac{n}{2}$.

Thus we solve $\xi_{t-1} - p_t\xi_{t-1} = \frac{n}{2}$ for p_t, which gives $p_t = 1 - \frac{n}{2\xi} = \frac{s}{n/2+s}$. The mean of the distribution describing the change in the value of a single unit is given by $\mu_1 = (-1)p_t + (0)(1-p_t) = -p_t$. The variance of this distribution is given by $\sigma_1^2 = (-1-\mu_1)^2 p_t + (0-\mu_1)^2(1-p_t)$, which simplifies to $\sigma_1^2 = p_t(1-p_t)$. Thus if the sum before the correction flip is ξ_t, so that ξ_t units participate in this flip, the variance of the $p_t(\xi_t|\xi_{t-1})$ distribution is $\sigma_{\xi_t}^2 = (p_t(1-p_t))\sqrt{\xi_t}$.

Substituting $p_t = 1 - \frac{n}{2\xi_t}$ into the expression for $\sigma_{\xi_t}^2$ yields

$$\sigma_{\xi_t}^2 = (\frac{n}{2\xi_t} - \frac{n^2}{4\xi_t^2})\sqrt{\xi_t}$$

after a line of algebra. We can estimate the algorithm's rate of convergence by considering the deterministic dynamics of a "typical value" of the sum ξ_t. For this typical value, we will use $\xi_t = \mu + \sigma_{\xi_t}^2$. (The distribution is non-stationary: the $\mu + \sigma_{\xi_t}^2$ mark is a moving target that by design converges to the mean). If the sum at time $t-1$ is ξ_{t-1}, then the value of ξ_t under the deterministic dynamics just described is

$$\xi_t = \frac{n}{2} + (\frac{n}{2\xi_{t-1}} - \frac{n^2}{4\xi_{t-1}^2})\sqrt{\xi_{t-1}} \tag{1}$$

Unfortunately, it is not straightforward to solve this non-linear difference equation for ξ_t as a function of n, t, and ξ_0. We can, however, model the dynamics of the algorithm by brute-force iteration of the difference equation.

Using $\xi_t = \frac{n}{2} + s_t$, we will first rewrite equation 2.5 in terms of s_t, the excess at time t. Some algebra yields

$$s_t = \frac{\sqrt{2} n s_{t-1}}{(n + 2s_{t-1})^{3/2}} \tag{2}$$

We assume that in the zeroth step, all the walkers are initially one, so that $s_0 = \frac{n}{2}$. This gives $s_1 = \frac{\sqrt{n}}{4}$ and $s_2 = \left(\frac{n}{2n+\sqrt{n}}\right)^{\frac{3}{2}}$. Further iterates become more complex. Iterating equation 2.5 numerically with $n = 10^6$, yields the following series of s values: $5.0 \times 10^5, 250, 0.353$. In simulations, the convergence was somewhat slower: typical excess values were $5.0 \times 10^5, 332, 1$. It might be possible to refine the analysis by convolving the sequence of complete probability distributions $p_t(s_t|s_{t-1})$, instead of using only the mean and variance to model the distribution. However, to the extent that the distributions are truly Gaussian, their higher moments should be 0. More likely the discrepancy between the analytical model and the simulation has to do with our choice of $\mu + \sigma_t^2$ as the "typical value" to study. Whatever its failings, the analysis is clearly better for large s than for small s. Below we will argue that the algorithm's dynamics enters a completely new regime when $s = 1$; presumably there is a continuous rather than discrete change to this new regime, and our analysis would ideally model that.

To study the scaling (as a function n) of an upper bound on ξ_t, we can linearise equation 2.5, ensuring that the linearised expression for ξ_t is always greater than the original. Since the expression $\frac{n}{2\xi_{t-1}} - \frac{n^2}{4\xi_{t-1}^2} = (p_t(1-p_t))$, the maximum value the expression can take is $\frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4}$. It is also the case that $\xi_t > \sqrt{\xi_t}$, so we can find an upper bound on the convergence time of ξ_t by solving the linear difference equation

$$\xi_t = \frac{n}{2} + \frac{1}{4}\xi_{t-1}$$

instead of equation 2.5. The solution of this linear difference equation turns out to be

$$\xi_t = \frac{n}{3}((\frac{1}{4})^t + 2)$$

The dynamics of the linearised equation are unfortunately somewhat different than the original, since ξ_t now converges to a value of $\frac{2n}{3}$ instead of $\frac{n}{2}$. Nevertheless, we will use the time to converge to the value $\frac{2n}{3} + 1$ as an (admittedly questionable) estimate of the convergence time of the first phase of the algorithm. Solving $\xi_t = \frac{2n}{3} + 1$ for t yields $t = \frac{1}{2}\log_2 n - \frac{1}{2}\log_2 3 < \log_2 n$. Thus

$$L_1 = \log_2 n$$

Endgame

Once the excess has been reduced to one, the algorithm's dynamics enters a new regime, since s_t cannot become any smaller than one without the algorithm terminating. Thus once the algorithm has reached this "endgame," equation 2.5 is no longer a good model (and the linearized version is even worse), so we must analyze it differently. The probability of exactly one of the $\xi = \frac{n}{2} + 1$ high units flipping in one time step (ie $\frac{n}{2} + 1$ trials) is given by a Poisson distribution: $w = \frac{\lambda^1}{1!}e^{-\lambda}$, where $\lambda = (\frac{n}{2} + 1)p$, with $p = \frac{s}{n/2+s} = \frac{1}{n/2+1} = \frac{2}{n+2}$. Writing out the expression for λ, one finds that it simplifies to one. Thus the probability of exactly one unit flipping in a single parallel trial is $\frac{1}{e}$.

In each iteration of the algorithm, either one unit flips, or some other number flips, so each iterate is a Bernouli trial with $p = \frac{1}{e}$. (If more than one unit flips, the units all revert to their previous state; thus there really are only two cases to worry about.) The probability of a certain number of trials L_2 occuring before success is given by a geometric distribution $p(L_2) = p(1-p)^{L_2 - 1}$. The probability of winning at any time l up to and including L_2 is therefore given by

$$p_s = \sum_{l=1}^{L_2} \frac{1}{e}(1 - \frac{1}{e})^{l-1} = 1 - (\frac{e-1}{e})^{L_2}$$

We have summed the geometric series to find the final expression. The probability of the algorithm failing to terminate prior to time L_2 is $p_f = 1 - p_s$. Solving for L_2, we find

$$L_2 = \frac{\ln p_f}{\ln \frac{e-1}{e}} = -2.18 \ln p_f$$

In order to make p_f less than 10^{-15}, we must chose $L_2 = 75$. The total running time L is given by

$$L = L_1 + L_2 = \log n + 2.18 \ln \frac{1}{p_f}$$

Under this algorithm, each unit needs $\log n$ bits of storage to hold its own ID, $\log n$ more to represent the random variable that it uses in deciding whether to take part in the corrective flip, and another $\log n$ to represent the ξ_t value that it reads from the AM bus. Thus the storage T needed for each unit is

$$T = \log n^3$$

Finally, the total number of random bits R used is the product of the number of trials L times $\log n^2$ (the number of storage bits per unit devoted to random variables; each of these gets randomized once per trial) times n, the number of units. $R = (\log n + 2.18 \ln \frac{1}{p_f})(\log n^2)n$ which can be rewritten

$$R = (n \log n^2)(\log n + 2.18 \ln \frac{1}{p_f})$$

3 Conclusion

The tables below summarize and compare the three algorithms. A point not illustrated by the tables is the fact that the three algorithms use different amounts of communication: in the first, symmetry breaking phase of the non-interacting algorithm, the units don't do any "joint" communication at all (the second phase of this algorithm, which does involve some communication, may be viewed as optional). In the sequential algorithm, the units learn the OR of the other units' random bits. In the parallel algorithm, the units learn the sum of the other units' random bits. Clearly a unit that learns the sum of its neighbors' random units has learned more than one that has only learned the OR of its neighbors' random bits. It would in fact be straightforward to determine the expected number of bits gained from the OR, AM-ADD, or other joint operations on the bus.

What is notable in examining the tables is that the algorithms that use more communication require less time. The parallel algorithm in particular requires far less running time than the others, and it is assumed to have much fatter commmunication pipes. One can imagine a framework analogous to the Shannon Channel Capacity formula for understanding the fundamental limits of symmetry breaking protocols: it would relate the number of units, the number of random bits required, the time to convergence, the communication rate, and probability of failure. Perhaps a way to begin would be to budget the entropy flow: entropy enters the system at the random number generator, then, through the communication mechanism, each unit learns some amount of information about the random variables in the other units. Finally, there is a stage in which entropy is rejected and IDs are assigned.

We have proposed and analyzed three distributed protocols for using a source of physical symmetry breaking to assign unique identities to a group of n processors. As the number of networked processors explodes in the next few years, distributed protocols that allow the processors to automatically and dynamically manage their own ID assignment may become increasingly appealing.

T (storage/unit)	
non-interacting	$\log_2 \frac{n^2}{p_f}$
sequential	$\log n$
parallel	$\log n^3$

L (time)	
non-interacting	$(n+1)\log_2 \frac{n^2}{p_f}$
sequential	$\frac{n \ln n}{\ln 2} + \log_2 \frac{1}{p_f}$
parallel	$\log n + 2.18 \ln \frac{1}{p_f}$

R (random bits)	
non-interacting	$n \log_2 \frac{n^2}{p_f}$
sequential	$n^2 \log 2n$
parallel	$(n \log n^2)(\log n + 2.18 \ln \frac{1}{p_f})$

T (storage/unit)	$n = 10^6$	$n = 320$
non-interacting	90	67
sequential	20	9
parallel	60	25

L (time)	$n = 10^6$	$n = 320$
non-interacting	9.0×10^7	2.1×10^4
sequential	2.0×10^7	2.7×10^3
parallel	89	81

R (random bits)	$n = 10^6$	$n = 320$
non-interacting	9.0×10^7	2.1×10^4
sequential	1.4×10^{13}	6.6×10^5
parallel	3.8×10^9	4.4×10^5

Acknowledgments

Many people contributed to the ideas in this paper. The original inspiration came from the Rhythm Tree, a network of 320 drum pads that was part of the Brain Opera, an interactive, multimedia opera conceived and directed by Tod Machover at the Media Lab. (The Brain Opera was in turn inspired by the work of Marvin Minsky.) Setting the drum pad IDs was a practical problem that had to be solved in the course of producing the opera.

Ara Knaian, who designed the Rhythm Tree hardware, originally suggested the "sequential symmetry breaking" algorithm, and included a noise circuit in the Rhythm Tree pad design. The theoretical work that lead to this paper was begun as a project in Yaneer Bar-Yam's Dynamics of Complex Systems class taught at MIT in the Spring of 1996. Yaneer and Isaac Chappell helped with the parallel symmetry breaking algorithm.

My advisor Neil Gershenfeld made useful suggestions about all aspects of the the project.

This work was supported by MIT Media Lab's Things That Think consortium, and a Motorola Fellowship.

References

[1] B. Awerbuch, L. Cowen, and M. Smith. Efficient asynchronous distributed symmetry breaking. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, Montreal, Quebec, Canada, 1994.

[2] E.W. Dijkstra. Solution of a problem in concurrent programming control. *Communications of the ACM*, 8(9):569, 1965.

[3] W. Heiligenberg. *Studies of Brain Function, Vol. 1: Principles of Electrolocation and Jamming Avoidance.* Springer-Verlag, New York, 1977.

[4] R.J. Lipton and A. Park. The processor identity problem. *Information Processing Letters*, 36:91–94, 1990.

[5] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, New York, 1995.

[6] J.A. Paradiso. The interactive balloon: Sensing, actuation, and behaviour in a common object. *IBM Systems Journal, http://www.almaden.ibm.com/journal/sj/mit/tocps.html*, 35(3 & 4), 1996.

[7] M.O. Rabin. The choice coordination problem. *Acta Informatica*, 17:121–134, 1982.

[8] Dallas Semiconductor. *Automatic Identification Data Book.* Dallas Semiconductor, Dallas, Texas, 1996.

[9] A.S. Tanenbaum. *Computer Networks.* Prentice-Hall, Englewood Cliffs, New Jersey, 1989.