AFRL-IF-RS-TR-2003-154
Final Technical Report
June 2003

BIO COMPUTING AND INFORMATION SYSTEMS: A QUEST

SUNY Institute of Technology

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

Tom Renz

APPROVED:
THOMAS E. RENZ
Project Engineer

James A. Collins

FOR THE DIRECTOR:
JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** JUNE 2003	**3. REPORT TYPE AND DATES COVERED** Final Aug 02 – Feb 03

4. TITLE AND SUBTITLE BIO COMPUTING AND INFORMATION SYSTEMS: A QUEST	**5. FUNDING NUMBERS** C - F30602-02-2-0194 PE - 61101E PR - BISC TA - AQ WU - 02
6. AUTHOR(S) Digen Das, Patrick Fitzgibbons, Larry Hash, Kamala Mahanta, and Nancy Bachman	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SUNY Institute of Technology Office of Sponsored Research PO Box 3050 Utica New York 13504-3050	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/IFTC 26 Electronic Parkway Rome New York 13441-4514	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2003-154

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Thomas E. Renz/IFTC/(315) 330-3423/ Thomas.Renz@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

This research project conducted an exploratory investigation of the viability of a Bio Computing and Information System, based on the available and up to date published scientific and technological information. The following three topics of research were identified as the research projects necessary for the development of a future Bio Computing and Information System:

1) Molecular Computation using Biological Cells: A Hybrid Approach.
2) The Bio-Safe Architecture: A Biologically inspired approach to the design of secure, scalable, adaptive and survivable Distributed Network Applications.
3) Development of a P- System with Active Membranes.

14. SUBJECT TERMS Biomolecular Computing, Membrane Computing	**15. NUMBER OF PAGES** 34
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

Table of Contents

Page

List of Figures

List of Tables

Summary

SUNYIT Institute of Technology, Utica, NY, in collaboration with SUNY College at Oneonta, NY, conducted an exploratory investigation of the viability of a Bio Computing and Information System, based on the available and up to date published scientific and technological information. The following four areas were identified for this investigation:

1) DNA Computing
2) Membrane Computing
3) Quantum aspects relevant to Bio Computing
4) The system aspects relevant to Bio Computing and Information System.

The study involved an extensive review of the current literature in the areas of interest mentioned above, critical analysis of the published results and their relevance to a future Bio System.

The outcome of this research has been presented in the following, in three segments corresponding to parts I, II, and III as mentioned in the Table of Contents.

PART I: Molecular Computation using Biological Cells: A Hybrid Approach

1.1 Introduction

Cells provide an isolated, controlled environment for carrying out complex chemical reactions. Moreover, they reproduce themselves, allowing the creation of many copies with little manufacturing effort. The ability to control cellular function will provide important capabilities in computation, materials manufacturing, sensing, effecting, and fabrication at the molecular scale. One particular short-term goal is to engineer chemical mechanisms which can be used to implement the digital abstraction--the notion that chemical signals can represent logical true and false (or zero and one) values.

Like any good abstraction, the digital abstraction allows us to ignore the fine details of a complex phenomenon, and concentrate on the essentials of the control process. The essential features of any digital logic implementation include the ability to distinguish and maintain two distinct values of some physical representation of a signal. This requires the presence of adequate noise margins--an ability to produce outputs whose physical values more perfectly represent a given logical value than the physical representation of their input. Adequate noise margins allow noise and imperfections in a digital system to be reduced, rather than amplified, during complex information processing.

What follows is a brief description of how this research effort should attempt to define a series of biologically plausible chemical reactions, which can be defined using a digital abstraction. A digital technology conventionally starts with Boolean logic gates, devices that operate on signals with two possible values, such as true and false, 1 and 0. An AND gate has two or more inputs and one output; the output is true only if all the inputs are true. An OR gate is similar except that the output is true if any of the inputs are true. The simplest of all gates is the NOT gate, which takes a single input signal and produces the opposite value as output: true becomes false, and false becomes true.

1.2 Building on Available Biological Mechanisms

In electronic circuits, a NOT gate can be made from a single transistor, wired so that a high voltage at the input produces a low voltage at the output, and vice versa. When the gate switches between its two states, it does so abruptly, like a snap-action light switch. It is this sudden, nonlinear response that gives digital devices their resistance to noise and error. Because a gate is either fully on or totally off, a signal can pass through a long chain of gates without degradation.

There are many biochemical equivalents to transistor gates. Perhaps the most interesting among them are the mechanisms of genetic control, which switch genes on and off. The archetypal example of genetic regulation in bacteria is the lac operon of *E. coli*, first studied in the 1950s by Jacques Monod and François Jacob. The lac operon is a set of genes and regulatory sequences involved in the metabolism of the disaccharide lactose. The bacterium's preferred nutrient is the simple sugar glucose, but when glucose is scarce, the cell can utilize lactose. The enzymes for digesting lactose are manufactured in quantity only when they are needed—specifically when lactose is present and glucose is absent.

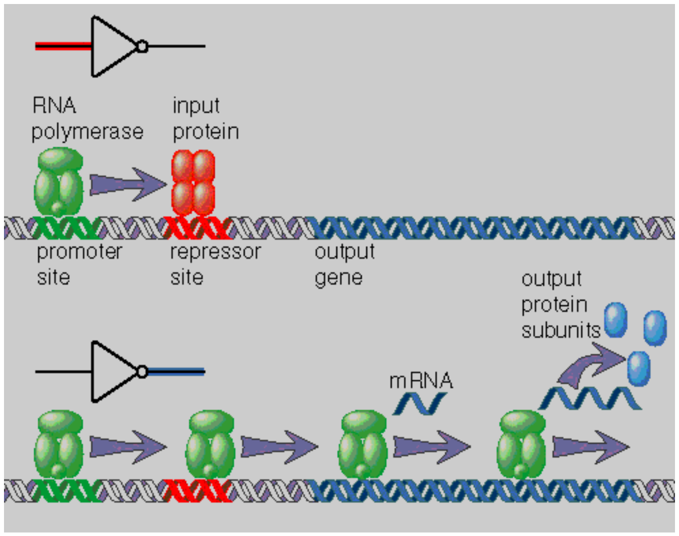


Figure 1 Synthesis of the lactose metabollic enzymes (Stage I)

Synthesis of the lactose metabolic enzymes is a two-stage process. First, the DNA is transcribed into messenger RNA by the enzyme RNA polymerase; then messenger RNA is translated into protein by ribosomes. The process is controlled at the transcriptional level. Before the genes can be transcribed, RNA polymerase must bind to the DNA at a specific site called a promoter, just "upstream" of the genes. Then RNA polymerase must travel along one strand of the double helix, reading off the sequence of nucleotides and assembling a complementary strand of messenger RNA. One mechanism of control prevents transcription by physically blocking the progress of the RNA polymerase molecule. This control is carried out by the lac repressor protein, which binds to the DNA downstream of the promoter region and blocks RNA polymerase action.

When lactose enters the bacterial cell, the lac operon is released from this restraint. A metabolite of lactose (an inducer) binds to the lac repressor, changing the protein's shape and thereby causing it to loosen its grip on the DNA. As the repressor protein drifts away, the polymerase is free to transcribe the operon.

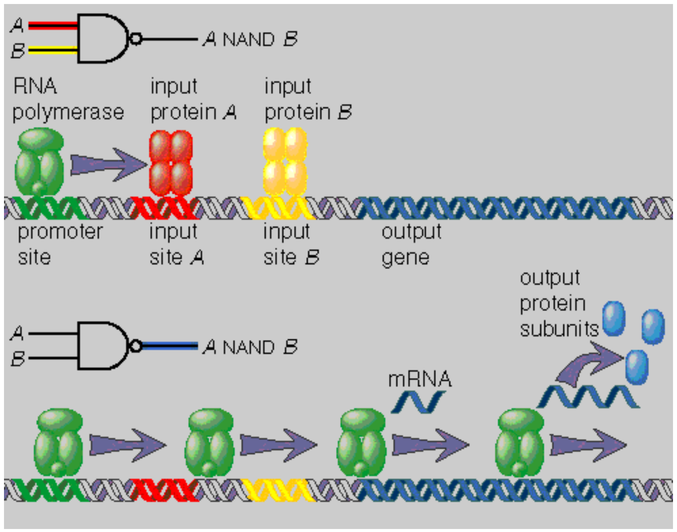


Figure 2 Synthesis of the lactose metabollic enzymes (Stage II)

The repressor system is only half of the lac control strategy. Even in the presence of lactose, the lac enzymes are synthesized only in trace amounts if glucose is also available in the cell. The reason, it turns out, is that the lac promoter site is a feeble one, which does a poor job of attracting RNA polymerase. To work effectively, the promoter requires an auxiliary molecule called catabolite activator protein, which clamps onto the DNA and makes it more receptive. Glucose causes the activator to fall away from the DNA just as lactose causes the repressor to let go—but the ultimate effect is the opposite. Without the activator, the lac operon lies dormant. All these tangled interactions of activators and repressors can be simplified by viewing the control elements of the operon as a logic gate. The inputs to the gate are the concentrations of lactose and glucose in the cell's environment. The output of the gate is the production rate of the three lac enzymes. The gate computes the logical function: (lactose and (not glucose)).

One of the questions that one needs to answer is whether these biochemical control mechanisms can exhibit the on-off, all-or-nothing character of digital circuits. Even if the transition between states is not perfect, the digital approximation must be satisfactory to be judged reliable. An underlying factor that tends to complicate matters is the cooperative action of multiple subunits in the regulatory proteins. The lac repressor consists of four subunits, and the catabolite activator protein has two. Although the first subunit may be slow in binding to the DNA, subsequent subunits interact with one another as well as with the DNA, so the binding goes faster. We will need to make certain that the threshold for repression or activation is stable.

The analogy between metabolic regulators and digital logic was already observed over 40 years ago. In 1961 Monod and Jacob wrote about genetic circuits and switching networks, and they described how activator and repressor proteins could be organized into systems that would function as memory elements and oscillators. Other researchers soon began exploring the connection between molecular biology and digital computing in greater depth and detail; for several years the theme was a frequent one in the *Journal of Theoretical Biology and the Bulletin of Mathematical Biophysics*.

The main focus of these early studies was on using digital models as a way of understanding events in the living cell. The Boolean approximation was a way of avoiding an unwieldy analysis of a complex chemical web. To follow all those molecular interactions in complete detail would have required tracking the concentrations of innumerable molecular species, measuring the rates of chemical reactions, and solving hundreds of coupled differential equations. By representing every gene as either being on or off reduced the problem to a simpler digital abstraction.

The biological computing model requires an antithetical approach to classical computing. Instead of constructing a computational model of biochemistry, one can exploit quasi-Boolean biochemistry to perform computation. This notion also has a history. In the 1970s Otto Rössler analyzed various coupled systems of chemical reactions that could implement the abstract computers called finite automata. More recently, other groups have looked at schemes of computing based on the catalytic activities of enzymes.

The most novel plan for biologically inspired computing was conceived by Leonard M. Adleman of the University of Southern California. His basic idea was to use the complementary base-pairing of DNA as a pattern-matching engine. Adleman himself and others have demonstrated the feasibility of this idea in experiments where vials of DNA carry out computational tasks in number theory and combinatorics.

All of the molecular computing methods mentioned above envision that the computation will be done in vitro. Although the molecules are of biological origin, they are extracted from the cell, and the reaction takes place in laboratory glassware. As a point of departure this research project is interested in exploring this from a different perspective- *in vivo* by turning the organism itself into a cellular logic gate, which in turn can be powered by its own metabolism.

1.3 A Biologically Plausible Gate

The first major goal of this research project is to develop design rules and a parts catalogue for biological logic gates, analogous to the comparable tools that facilitate design of electronic integrated circuits. An engineer planning the layout of a silicon chip does not have to define the geometry of each transistor individually; those details are specified in a library of functional units, so that the designer can think in terms of higher-level abstractions such as logic gates and registers. A similar design discipline will be needed before biocomputing can become practical.

The elements of this research project's bio computing design library will likely consist of repressor proteins and their target DNA binding sites. The logic "family" might be named RRL, for repressor-repressor logic, in analogy to the long-established TTL, which stands for transistor-transistor logic. The basic NOT gate in RRL will be a gene encoding some repressor protein (call it Y), with transcription of the Y gene regulated in turn by a different repressor (call it X). Thus whenever X is present in the cell, it binds near the promoter for Y and blocks the progress of RNA polymerase. When X is absent, transcription of Y proceeds normally. Because the Y protein is itself a repressor, it can serve as the input to some other logic gate, controlling the production of yet another repressor protein, say Z. In this way gates can be linked together in a chain or cascade. Transcriptional output can be measured as levels of a reporter protein, such as green fluorescent protein (GFP) or as activity from reporter enzymes, such as luciferase or β-galactosidase; these would comprise additional modular elements of the logic gates.

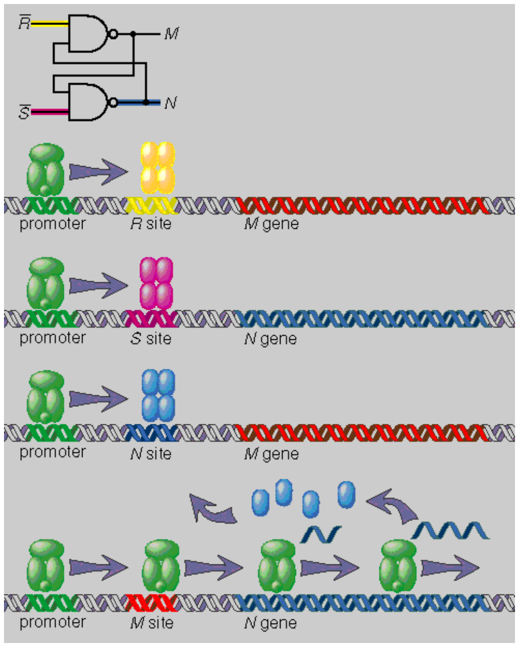


Figure 3 Concepts of Biological Switches

Genetic engineering tools such as polymerase chain reaction (PCR) amplification of particular repressor or reporter genes, chemical synthesis of oligonucleotides encoding regulatory elements or adapter regions, and ligation into plasmid vectors, can be used to experimentally assemble the components. Additional flexibility in the design and execution of molecular logic gates comes from the ability to introduce plasmids with various genetic elements into bacteria via transformation or electroporation. Gates can be sequentially controlled by introducing components on separate plasmids with compatible replicons. Further control of bacterial logic gates is facilitated by small molecular inducers (for example IPTG for the lactose repressor) or temperature sensitive forms of some repressors. Well characterized *E. coli* bacterial strains, with suitable genetic and physiological properties to serve as hosts for biocomputing, are readily available.

Going beyond the NOT gate to other logical operations is inherently more complex. Inserting binding sites for two repressor proteins (A and B) upstream of a gene for protein C creates a NAND gate, which computes the logical function "NOT AND".

With the dual repressor sites in place, the C gene is transcribed only if both A and B are absent from the cell; if either one of them should rise above a threshold level, production of C stops. It is a well-known result in mathematical logic that with enough NAND and NOT gates, it is possible to generate any Boolean function required. For example, the function (A or B) is equivalent to (not (A nand B)), while (A and B) is ((not A) NAND (not B)). The NOT gate itself can be viewed as just a degenerate NAND with only one input. Thus with just a collection of NAND gates, it is possible to construct any logical network.

Pairs of NAND gates can also be coupled together to form the computer memory element known as a flip-flop, or latch. Implementing this concept in RRL calls for two copies of the genes coding for two repressor proteins, M and N. These could be introduced into the same cell by transformation or electroporation of plasmid vectors with compatible replicons. One copy of the M gene is controlled by a different repressor, R, and likewise one copy of the N gene is regulated by repressor S. The part comes in the control arrangements for the second pair of genes: Here the repressor of M is protein N, and symmetrically the repressor of N is M. In other words, each of these proteins inhibits the other's synthesis. The flip-flop can be envisioned as follows: assume initially that both R and S are present in the cell, shutting down both of the genes in the first pair; but protein M is being made at high levels by the M gene in the second pair. Through the cross-coupling of the second pair, M suppresses the output of N, with the collateral result that M's own repressor site remains vacant, so that production of M can continue. But now imagine that the S protein momentarily falls below threshold levels, perhaps by binding to an inducer molecule. This event briefly relieves the repression of the N gene in the first pair. The resulting pulse of N protein represses the M gene in the second pair, lowering the concentration of protein M, which allows a little more N to be manufactured by the second N gene, which further inhibits the second M gene, and so on. Thus a momentary change in S switches the system from steady production of M to steady production of N. Likewise a brief change in R would switch it back again. (S and R stand for "set" and "reset.")

Some of the possible complexities of molecular gates can be seen in the modular regulatory networks developed by Guet and coworkers. Using vectors containing modular repressors and their target control regions driving a green fluorescent protein reporter, they have developed bacterial strains that parallel the logic functions of NOR, NAND, and NOT IF in response to simple molecular signals.

One conclusion to be drawn from this synopsis of a few RRL devices is that a computer based on genetic circuits will need a sizable repertory of different repressor proteins. Each logic gate inside a cell must have a distinct repressor assigned to it, or else the gates would interfere with one another. In this respect, a bio molecular computer is very different from an electronic one, where all signals are carried by the same medium—an electric current. The reason for the difference is that electronic signals are steered by the pattern of conductors on the surface of the chip, so that they reach only their intended target. The biological computer is a wireless device, where signals are broadcast throughout the cell. The need to find a separate repressor for every signal complicates the designer's task, but there is also a compensating benefit. On electronic chips, communication pathways claim a major share of the space. In a biochemical computer, communication comes without such an overhead cost.

1.4 Implementation Issues

Any new digital logic family has important characteristics and limitations, which must be understood in order to effectively design with them. In this section, we have attempted to predict some of these issues, but inevitably there will be many which we will overlook until real implementation of these gates is underway.

Perhaps the most dramatic difference between our biological gates and conventional logic gates is the difference in speed. Electrical gates now function with delays of tens of picoseconds. Biological gates constructed using this methodology will have delays governed by the speed of protein manufacturing--perhaps many minutes. Roughly speaking, we should think of this logic family as functioning at frequencies measured in millihertz, rather than at rates measured in Megahertz. While other biological mechanisms might be constructed which could optimistically function at kilohertz rates, such structures will require a degree of engineering design which we believe will not be available in the short term.

A critical resource in the design of complex logic circuits within a cell is the availability of a sufficient number of distinct DNA binding proteins. Not only must many such proteins be found, but the set employed must *not* be used elsewhere within the host cell control mechanisms. In *E. coli*, numerous possible repressors could be tested. Pérez-Rueda and Collago-Vides used bioinformatic analysis of the *E. coli* genome to estimate that about 314 different DNA-binding transcriptional regulators exist; of these 42% (about 135) are expected to act as repressors and are potentially adaptable as components of molecular logic gates. About a dozen bacterial repressor proteins have been characterized in atomic detail and these would serve as the initial repertoire of molecular components.

Protein design, and, especially, protein-complex design, required for such high-speed gates, is not sufficiently well understood at the present time. Yet another limitation comes from the requirement of finite concentrations of these proteins within the cell. Cells have finite volume, and the requirement that many copies of a protein are needed to have an effect, together with the complexity requirement for many distinct proteins, leads to an upper limit on the logic complexity which can be performed within a single cell. We are confident, however, that logic circuits of an interesting level of complexity can be constructed.

Another potential problem is the repressor molecules taking part in the computation must also be distinct from those involved in the normal metabolism of the cell. Otherwise, a physiological upset could lead to a wrong answer; or, conversely, a computation might well “poison” the cell in which it is running. A toxic instruction might actually be useful—any multitasking computer must occasionally "kill" a process—but unintended events of this kind would be involve extensive and perhaps an unacceptable amount of debugging. Unlike the silicon computer, it is not possible to just “reboot” a dead bacterium. Yet we can see that nature faces the same problem: A multitude of metabolic pathways have to be kept under control without unwanted crosstalk. As a result, cells have evolved thousands of distinct regulatory proteins. Moreover, the biocomputing engineer will be able to mix and match among molecules and binding sites that may never occur together in the natural world. The aim of the RRL design rules is to identify a set of genes and proteins that can be encapsulated as black-box components, to be plugged in as needed.

The research project should use a design simulator, which allows a device to be tested without the substantial effort of building a prototype. The world of electronics has long relied on a simulator called Spice, which models the physics of transistors and other electronic components. The research project may use the BioSpice simulator, which can be used to model the dynamics of genetic circuits in a similar way. It is important to note, however, that this research project will not solely base its design work on such simulations, but will also include some "wet" experiments. For example this may include experimenting with a free-running genetic oscillator by arranging repressor genes so that they act on one another. If we can find periodic fluctuations in gene expression, with a frequency independent of the cell's reproductive cycle it would allow for a "genetic toggle switch" much like the flip-flop described above, with two cross-coupled promoters and repressors. The eventual aim of the project is the construction of "genetic applets"—self-contained program modules that could be "downloaded" into organisms.

It is understood that designing "cell gates" which are necessary to perform useful computations is not a trivial undertaking. The overriding objective is to produce cells working in concert on the same task. From the biological point of view, mass producing bacteria is extraordinarily easy and it does not require a multimillion dollar "fab line" to manufacture them. The difficult part is organizing a population of cells so that they work toward some specified goal. Here again electronic and biological technologies diverge.

On a silicon chip, every circuit element has an assigned place and function, but living cells are squishy and motile and not easily confined to a rigid grid. The project should plan to take another cue from biology by experimenting with large-scale structures which emerge from natural phenomena, such as cell to cell signalling. Bacteria might be organized into higher order structures by means of short-range communications between neighboring cells and the diffusion of chemical signals over larger distances. Thus it might be possible to fabricate a molecular chip consisting of independent microorganisms that work as a functional computing unit.

The study of large arrays of simple processing elements is a classical topic in computer science, but for the most part the arrays have been geometrically regular, and the processors have operated in strict synchrony. The research project should try to apply the new paradigm of "amorphous computing" which has been used successfully to design spatially irregular and unsynchronized arrays. It is interesting to note that from the results generated so far these arrays have a distinctly botanical look to them, and yet they also resemble the design drawings for a silicon integrated circuit.

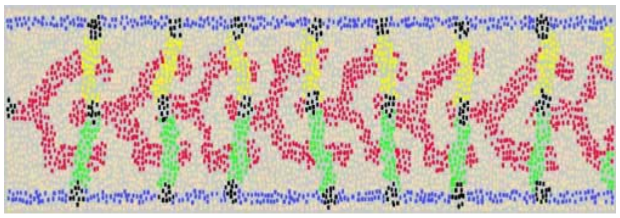


Figure 4 Spatially irregular and unsynchronized arrays

1.5 Applications

Cellular computing opens a new frontier of engineering that will dominate the technology of the next century. Employing information technology, the future holds promise for the development of means to organize and control biological processes that are just as effective as our current mastery of electrical processes.

In particular, biological cells are self-reproducing chemical factories that are controlled by a program written in the genetic code. Current progress in biology will soon provide us with an understanding of how the code of existing organisms produces their characteristic structure and behavior. As engineers we can take control of this process by inventing codes (and more importantly, by developing automated means for aiding the understanding, construction, and debugging of such codes) to make novel organisms with particular desired properties.

Besides the obvious application of control of biological processes to medicine, we will be able to co-opt biological processes to manufacture novel materials and structures at a molecular scale. The biological world already provides us with a variety of useful and effective mechanisms, such as flagellar motors. If we could co-opt cells to build organized arrays of such motors, with accessible interfaces for power and control, we could see how this could be of engineering significance. Common, biologically available conjugated polymers, such as carotene, can conduct electricity, and can be assembled into active components. If we, as engineers, can acquire mastery of mechanisms of biological differentiation, morphogenesis, and pattern formation, we can use biological entities of our own design as construction agents for building and maintaining complex ultramicroscopic electronic systems. Such systems will have better performance and reliability then technologies based on less precisely controlled chemical processes. Of course, one of the most important products of mass-produced molecular-scale engineering will be extremely compact, efficient, and effective computing mechanisms.

Thus, in spite of the long gate delays in cellular computing mechanisms, the fact that cells can reproduce and organize into precisely arranged and differentiated tissues means that we can use them as the (very slow) agents of molecular-scale manufacturing of macroscopic objects. It is the resulting objects that we desire--they may contain electrical circuitry with picosecond cycle times. The biological systems are our fabrication assembly, with proteins as the building block, and with DNA as our programming mechanism (see Figure 5).

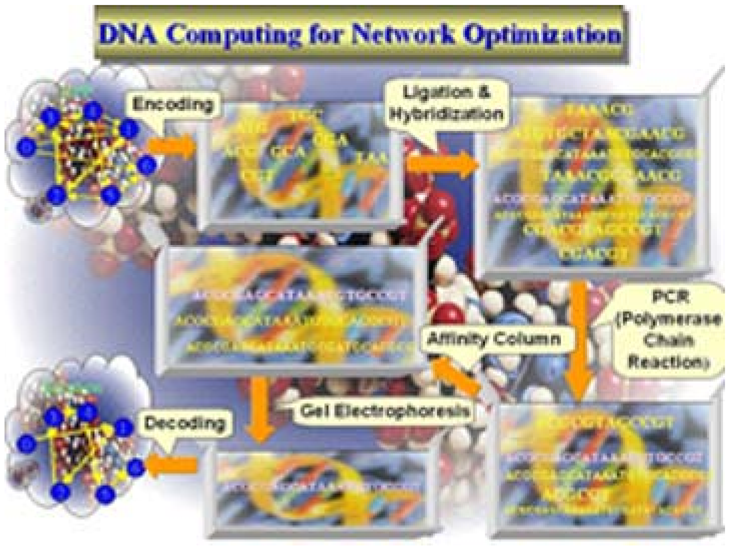


Figure 5 DNA Computing for Network Optimization

1.6 Quantum Aspects of Bio-systems

As described above, the physical processes occurring in classical electronic systems and bio-systems exhibit, in their computational characteristics, a significant degree of analogy even though the processes are completely different. The gate and switch functions that shape the world of computation are realizable in both systems, showing that bio-computers are as feasible as the silicon computers so inseparably tied to our technology-driven life style.

Parallel to the development of the computational ability of bio-systems such as DNA and Membrane systems, the search for the ultimate security and speed of information dissemination has led to the vigorous research on quantum information that we see around us today.

The classical bits 0 and 1 used in classical computation are two distinct states analogous to the two sides of a coin. A quantum bit or “qubit” on the other hand can exist in a “continuum” of states between $|0\rangle$ and $|1\rangle$, which collapses to 0 or 1 when measured. To state more specifically, it can be in a state such as $(1/\sqrt{2})\,|0\rangle + (1/\sqrt{2})\,|1\rangle$ with a 50% probability $(1/\sqrt{2})^2$ of giving the result 0 and 50% probability $(1/\sqrt{2})^2$ of giving the result 1. This “impossible to witness, but available to compute 0 and 1 in many different ways” existence for qubits enables the computing process to have much greater options. Practical examples include the spin states of the atom, the polarization states of the photon, the orbital angular momentum states of the photon. The biological bits, for the time being, seem to have equivalence with the classical situation in that the processes involved therein enable the ability to distinguish and maintain two distinct values of some physical representation of a signal as required for digital abstraction. Does the biological process exhibit any of the continuum effect observed in the quantum information process?

Gates and switches are the building blocks of computers and those need to be investigated for the biological processes. The simplest NOT gate is feasible in all three cases. A comparison shows only one possibility for classical systems, whereas there are many possibilities for a quantum NOT gate. It can be summed up very nicely with the statement "Any unitary matrix specifies a valid quantum gate" which means that, contrary to the existence of only one non-trivial single bit classical gate ($0 \rightarrow 1$ or $1 \rightarrow 0$), there exist many non-trivial single qubit gates such as the Z-gate

$$Z \equiv \begin{vmatrix} 1 & 0 \\ 0 & -1 \end{vmatrix}$$ which leaves $|0\rangle$ unchanged and flips the sign of $|1\rangle$ to yield $-|1\rangle$, and,

the Hadamard Gate H

$$H \equiv (1/\sqrt{2}) \begin{vmatrix} 1 & 0 \\ 0 & -1 \end{vmatrix}$$

which turns a $|0\rangle$ into $(|0\rangle + |1\rangle)/\sqrt{2}$ (first column of H), halfway between $|0\rangle$ and $|1\rangle$, and turns $|1\rangle$ into$(|0\rangle - |1\rangle)/\sqrt{2}$ (second column of H), which is also halfway between $|0\rangle$ and $|1\rangle$. Applying H twice does not change the state at all, so H^2 is not a NOT gate.

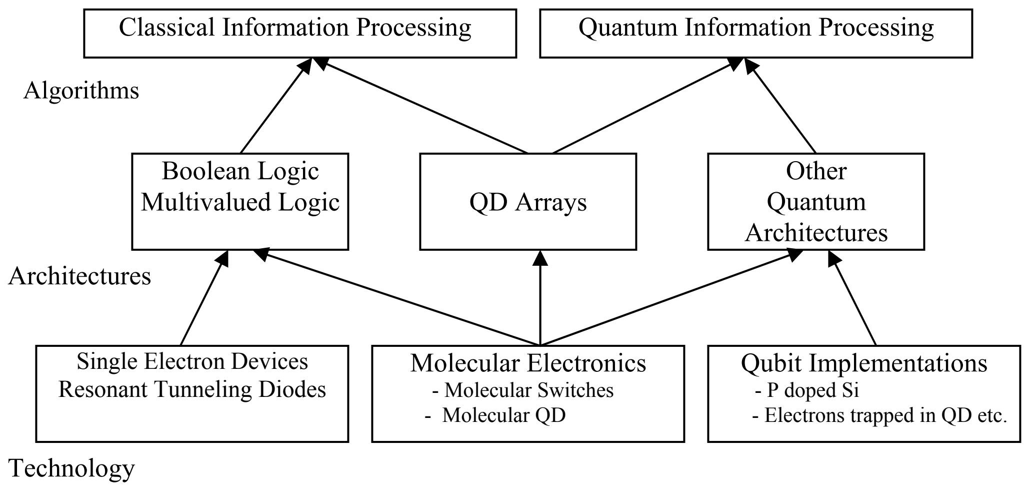
How does the biological NOT gate fit into this picture? Do we look for greater versatility in it than what the classical gate offers? Is a hybrid system likely to exhibit some of the qualities expected of a quantum system?

The reversible nature of the quantum phenomena (until measured?) has led to the Controlled NOT (CNOT) gate, such as the Toffoli and Fredkin reversible gates, for quantum computation using multiple qubits to perform in the roles of all the classical gates AND, OR, XOR, NAND, and NOR gates.

Fault tolerance is another issue in both classical and quantum information; a host of error-correction techniques have been designed. How do biological circuits perform in a noisy environment?

1.7 Architectures for mixed bio/quantum computing based on QCA model:

The research project proposes a mixed methods approach, which includes biological/quantum computation and is based on the Quantum Cellular Automata (QCA) model (see Figure 6). The concept of mixed biological/quantum architecture relies on the possibility of implementing Boolean logic as well as quantum logic by using cells. An example of an expectation from such a process would be coupling of the quantum registers with a biological network consisting of cells. The quantum/biological coupling being equivalent to a measurement, it must be controlled to ensure that the presence of the biological network is non-invasive while the quantum operations are being performed and efficient enough after their completion.



Classical Information Processing
Quantum Information Processing
Algorithms
Boolean Logic Multivalued Logic
QD Arrays
Other Quantum Architectures
Architectures
Single Electron Devices Resonant Tunneling Diodes
Molecular Electronics - Molecular Switches - Molecular QD
Qubit Implementations - P doped Si - Electrons trapped in QD etc.
Technology

Figure 6: An overview of the devices based on quantum effects

Since any form of computing is basically a physical process which needs the laws of the nano-world for clear explanation, it is obvious that any attempt on a bio-computer or a hybrid thereof will ultimately need some sort of quantum analysis. This has been the driving factor behind the inclusion of the study of the quantum aspects of bio-systems.

The proposed research should endeavor to find answers to the host of questions yet to be addressed, especially regarding a hybrid computing and information system.

Part II: The Bio-SAFE Architecture

2.1 Introduction

It is not difficult to imagine a future where billions of people regularly access applications running inside the global network as part of their daily lives. To make this future a reality, network applications must overcome three critical challenges. First, they must scale to handle the enormous demand placed upon them. Second, they must adapt to dynamic user demand and network conditions. Finally, network applications must survive partial failures and remain available to their users yet be secure enough to prevent outside attacks from hackers and crackers.

Over millions of years of evolution, large scale biological systems, such as the bee or ant colony, have developed mechanisms that allow them to scale, adapt, and survive. Consider the bee colony [17]. Bee colonies scale to a large number of bees because all activities of the hive are carried out without centralized control. Bees act autonomously, influenced by local conditions and local interactions with other bees. When building the hive, bees are guided only by the structure of the partially completed hexagonal cells around them. There is no master bee that controls the building of the hive. The bee colony also adapts to dynamic conditions, often to optimize its food gain relative to energy expenditure. When the amount of honey in the hive is low, a large number of food gathering bees leave the hive to gather nectar from the flowers in the area. When the hive is nearly full of honey, most bees remain in the hive and rest. The bee colony is survivable because it is not dependent on any single bee, not even the queen bee. Therefore, the colony can still survive after a predator kills a number of bees. In fact, the desirable characteristics of the bee colony, scalability, adaptability, and survivability, are not present in any single bee. Rather, they emerge from the collective actions and interactions of all the bees in the colony.

We believe that the challenges faced by future network applications have already been overcome in large-scale biological systems and that future network applications will benefit by adopting key biological principles and mechanisms. Our initial effort at applying classical network principles and mechanisms to the design and implementation of highly secure, distributed network applications has produced the Secure Architecture for Extensible Mobile Internet Transport Systems (SAFEMITS) Architecture. The SAFEMITS Architecture is essentially a "blueprint" as well as middleware for the design and implementation of highly secure, scalable, adaptive, and survivable/available network applications. We are proposing extending the concept of SAFEMITS to using a biological system design which we have termed the Bio-SAFE Architecture. The Bio-SAFE Architecture will be based on the principles and mechanisms that allow biological systems to scale, adapt, and survive. The research objective is too provide a general model that guides the design of scalable, adaptive, and survivable/available network applications. The middleware of the SAFEMITS Architecture consists of software components that aid the implementation of network applications. The middleware implements critical aspects of the paradigm so that applications implemented using the middleware will conform to the paradigm of the Bio-SAFE Architecture.

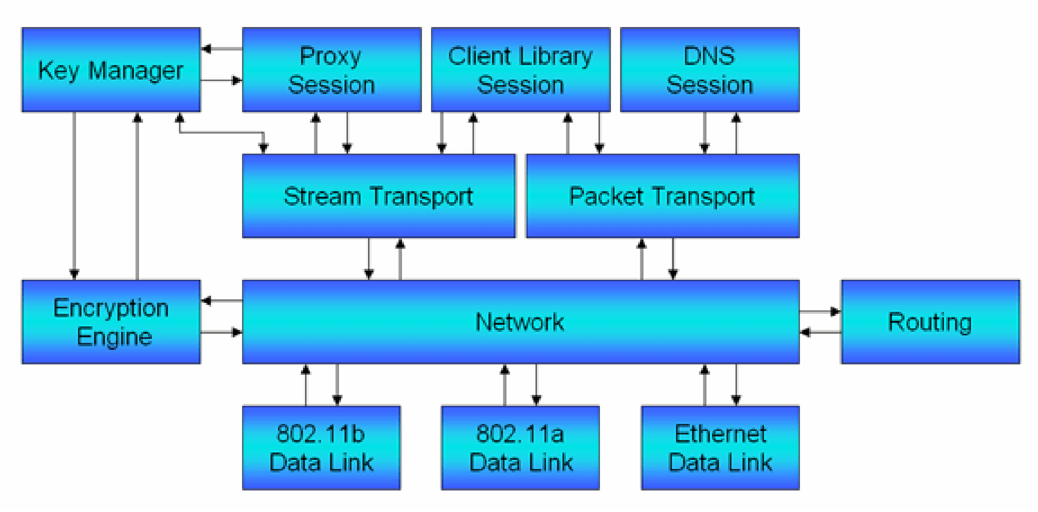


Figure 7. SAFEMITS Architecture Diagram

Using the SAFEMITS Architecture, we have successfully demonstrated a distribution application called Battle Space and simulated its performance. We believe it is possible to extend the potential for SAFEMITS through the use of the biologically inspired approach. We continue to simulate, design, and implement other applications using the SAFEMITS Architecture in order to increase our understanding of the power and limitations of this approach.

2.2 Bio-SAFE Architecture

The Bio-SAFE architecture is a paradigm as well as middleware for the design and implementation of scalable, adaptive, and survivable/available network applications. The paradigm is based on the principles and mechanisms that allow biological systems to scale, adapt, and survive. While the paradigm guides the design of a network application, the middleware aids the implementation of the application by providing software components, namely mobile entities or ME's and Bio-SAFE platforms. Mobile-entities are autonomous mobile agents that are used to implement network applications. Bio-SAFE platforms provide execution environments and support services for the mobile-entities. Section 2.3 describes the key biological principles and mechanisms that are the basis for the paradigm of the Bio-SAFE Architecture. Section 2.4 describes the middleware of the Bio-SAFE architecture.

2.3 A biological paradigm

After surveying different types of large-scale biological systems [4], [5], [7], [8], [16], [17], we have extracted a number of principles and mechanisms that enable them to scale, adapt, and survive[1]. In the paragraphs below, we describe these biological principles and mechanisms and show that they are the basis for the paradigm of the Bio-SAFE Architecture.

Emergence. Many desirable characteristics of large-scale biological systems, including scalability, adaptability, and survivability, are not present in any single biological entity. Instead, desirable characteristics emerge from a group of interacting biological entities. Therefore, in the Bio-SAFE Architecture, applications are implemented using a collection of autonomous mobile agents, called mobile-entities (described in Section2.2.2). The desirable characteristics of applications, such as scalability, adaptability, and survivability/availability, emerge from the collective actions and interactions of their constituent mobile-entities.

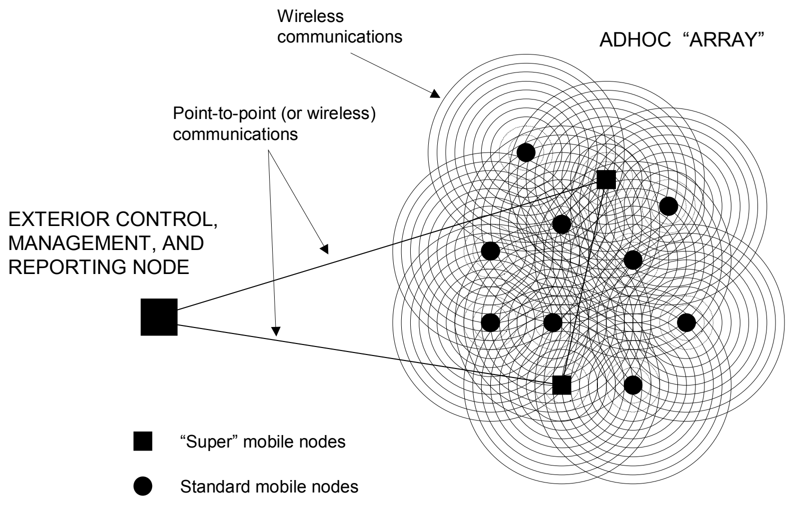

Wireless communications
ADHOC "ARRAY"
Point-to-point (or wireless) communications
EXTERIOR CONTROL, MANAGEMENT, AND REPORTING NODE
■ "Super" mobile nodes
● Standard mobile nodes

Figure 8: Distributed mobile network architecture

Autonomous actions based on local information and local interactions. Individual biological entities in large-scale biological systems act autonomously. They may be influenced by conditions in their local environment or by interactions with a limited number of other biological entities; however, there is no "master" entity that collects information and controls the actions of others. In the Bio-SAFE paradigm, mobile-entities also act autonomously based on local information and local interactions with other mobile-entities. In the Bio-SAFE paradigm, "local" is defined as both spatially adjacent and numerically limited. For example, a Mobile-entity may obtain information regarding the Bio-SAFE platforms that are adjacent to the Bio-SAFE platform it is currently on (spatially adjacent locality) and information regarding a limited number of other Bio-SAFE platforms that may be in distant parts of the network (numerically limited locality). Similarly, a Mobile-entity can interact with other mobile-entities on the same Bio-SAFE platform as itself (spatially adjacent locality) and with a limited number of other

[1] We are continuing our study of large scale biological systems in order to find other principles and mechanisms. However, we believe the biological principles and mechanisms that we have extracted thus far provide an adequate basis for our initial work.

mobile-entities on possibly distant Bio-SAFE platforms (numerically limited locality). We believe that autonomous mobile-entities using local information and local interactions greatly enhance the scalability of an application because they may be replicated throughout the network.

Birth and death as expected events. Biological entities regularly die from various causes, e.g. starvation, old age, or predation. However, the biological system survives because the death of biological entities is compensated by the birth of new biological entities. In the Bio-SAFE Architecture, applications are also survivable because they are implemented using multiple mobile-entities that can replace each other. When a Mobile-entity crashes or dies, other mobile-entities may reproduce to maintain the Mobile-entity population. As a result, the application remains available to users.

Energy and adaptation. In the biological world, biological entities adapt their actions to maximize their energy gain while minimizing their energy expenditure. Biological entities that cannot collect enough energy to support their normal life functions will die of starvation. Biological entities that are able to collect and store an abundance of energy are more likely to reproduce. In the Bio-SAFE Architecture, mobile-entities collect energy[2] from human users or other mobile-entities. They must also give energy to Bio-SAFE platforms (described in Section 2.4.1) in order to execute. Like biological entities, mobile-entities also try to maximize their energy gain while minimizing energy expenditure. If a mobile-entity exhausts its energy units, it will not be allowed to execute by the Bio-SAFE platform, i.e. it dies of starvation. Mobile-entities with an abundance of stored energy are also more likely to reproduce. Because energy plays an important role in the lifecycle of mobile-entities, mobile-entities (and hence, the applications that they form) are forced by energy considerations to adapt to user demand and the network environment.

Natural selection and evolution. In the biological world, evolution occurs as a result of genetic diversity (created by crossover and mutation) and natural selection. Crossover, mutation, natural selection, and evolution are also present in the Bio-SAFE Architecture. When a Mobile-entity reproduces with another Mobile-entity, their behaviors (described in Section 2.4.2.2) randomly recombine (crossover) to form the child's behaviors. The child's behaviors may also contain mutations. Natural selection occurs because mobile-entities that exhaust their energy units are not allowed to execute (they die), and mobile-entities with an abundance of energy units are more likely to reproduce. Therefore, if a mobile-entity's behaviors enable it to collect and store more energy than other mobile-entities (implying that it is more useful and/or efficient), it will live longer and give birth to more mobile-entities similar to itself. Over time, the mobile-entities that comprise an application will evolve to behaviors that are more useful and/or efficient.[3]

[2] In the Bio-SAFE Architecture, energy is equivalent to money. However, the Bio-SAFE Architecture does not depend on a heavy weight electronic cash system to handle the energy/money or to prevent cheating. There are light weight techniques that reduce the incentive to cheat or reduce the risk of fraud. For example, energy units can be made almost worthless (e.g. 1 energy unit = 1/10000 of a cent). Strong security measures are used on large transactions only. Energy units can also represent something valuable (e.g. frequent flyer miles, access to services or resources) and yet be not redeemable for money. A complete discussion of light weight techniques to handle energy/money in the Bio-SAFE Architecture is beyond the scope of this research.
[3] Because random crossover and mutation may require a long period of time to "discover" useful/efficient behaviors, human designers may insert mobile-entities with new, useful/efficient behaviors into the network to increase the speed of evolution.

2.4 The biological middleware

The previous subsection described the biological principles and mechanisms that are the basis for the paradigm of the Bio-SAFE Architecture. This subsection describes the middleware which we are interested in developing for the Bio-SAFE Architecture. The Bio-SAFE middleware is necessary to implement critical aspects of the paradigm so that network applications implemented using the middleware will conform to the paradigm of the Bio-SAFE Architecture. The middleware as envisioned will consist of two components, the Bio-SAFE platform and the mobile-entity, which are described below.

2.4.1 Bio-SAFE platform

Bio-SAFE platforms will provide execution environments and support services for mobile-entities. Our definition of a Bio-SAFE platform is any networked hardware device that provides a virtual machine interface (e.g. the Java Virtual Machine) and will run the Bio-SAFE platform software. The Bio-SAFE platform software will contain functionality and services not present in the virtual machine. These functionality and services are described below.

Resource Control: The Bio-SAFE platform software will provide resources, e.g. CPU time, memory, disk space, network bandwidth, to mobile-entities in exchange for energy units. When a Mobile-entity is created/born, it is given energy units by the system administrator who created it or by its parent mobile-entities. As explained in Section 2.3, a Mobile-entity may receive additional energy units from users or other mobile-entities during its lifetime. Mobile-entities use their energy units to secure resources from the Bio-SAFE platform. If a Mobile-entity exhausts its energy units, it will not be allowed to execute on the platform and the disk and memory resources that contain its code and data will be freed, i.e. the Mobile-entity dies of starvation. By “killing” mobile-entities when they exhaust their energy units, the platform software contributes to the process of natural selection in the network environment, which is a critical aspect in the paradigm of the Bio-SAFE Architecture.

Mobile-entity Scheduling: Assuming that a Mobile-entity has purchased sufficient resources, the platform software schedules the Mobile-entity for execution when an event of interest to the Mobile-entity occurs. For example, the Mobile-entity will be scheduled for execution when a message for the Mobile-entity arrives. If the Mobile-entity set a timer using the timer system service, then the Mobile-entity will be scheduled for execution when the timer expires.

System Services: The platform software will provide system level services that mobile-entities cannot perform directly, for example, migration, reproduction, and event notification. Since these services consume CPU, memory, and/or network resources, mobile-entities must pay additional energy units to the platform software to receive these services.

Information Services: The platform software will also provide information regarding the local environment to all mobile-entities on the platform. Some examples the information provided by the platform software will be the location of the platform, a list of mobile-entities on the platform, and a list of neighboring Bio-SAFE platforms to which the Mobile-entity can migrate. Consistent with the paradigm of the Bio-SAFE Architecture, the platform software will not provide global information, such as a list of all other mobile-entities in the network or the locations of all other Bio-SAFE platforms.

2.4.2 Mobile-entity

The mobile-entity is an autonomous mobile agent, analogous to an individual bee in a bee colony. In the Bio-SAFE Architecture, multiple interacting mobile-entities are used to implement an application. The Mobile-entity can be logically organized into attributes, behaviors, and body. These three parts are described below.

2.4.2.1 Attributes

Attributes are variables that describe the Mobile-entity. Some examples are *ownerName*, *uniqueID*, *timeBorn*, and *energyLevel*. These variables have remotely accessible interfaces that allow human users or other mobile-entities to read their values. *OwnerName, uniqueID, timeBorn,* and *energyLevel* are a very small sample of the attributes that a Mobile-entity may have. We believe that creating a rich set of attributes is critical to enabling and promoting interesting interactions among mobile-entities.

2.4.2.2 Behaviors

Mobile-entity behaviors consist of executable code that implement the functionality of a Mobile-entity and control its autonomous actions. Mobile-entities may have arbitrary numbers of behaviors. In this subsection, we first describe some possible Mobile-entity behaviors. We then describe the internal structure of Mobile-entity behaviors and explain how this structure allows mobile-entities to adapt and evolve in the network environment. We conclude this subsection with an extended example of a Mobile-entity behavior.

Below are descriptions of some possible Mobile-entity behaviors. It is entirely possible that upon experimentation we will discover that Mobile-entities may have behaviors different from those described below.

Receive Event: The Bio-SAFE platform software will invoke this Mobile-entity behavior when an event of interest to a Mobile-entity occurs. The Receive event behavior then invokes other Mobile-entity behaviors to process the event. For example, if the event is “received message”, then the Receive event behavior invokes the Receive message behavior (described below) to do further processing of the message. If the event is “timer expired”, then this behavior invokes the set timer, expend energy, migration, reproduction, relationship, and death behaviors.

Set Timer: This behavior enables a Mobile-entity to request a timer event notification from the Bio-SAFE platform software. This behavior ensures that mobile-entities will execute on a periodic basis so that they may decide whether they want to perform certain actions, such as migration or reproduction.

Expend Energy: This behavior enables a Mobile-entity to use its energy units to buy resources from the Bio-SAFE platform software.

Migration: This behavior enables a Mobile-entity to determine whether it should migrate and which Bio-SAFE platform to migrate to. When a Mobile-entity decides to migrate to another Bio-SAFE platform, it invokes the migration service of the Bio-SAFE platform to perform the actual migration.

Reproduction: Mobile-entities can reproduce sexually or asexually. This behavior enables a Mobile-entity to determine whether it should reproduce, and in the case of sexual reproduction, which Mobile-entity to reproduce with. When a Mobile-entity decides to reproduce, it invokes the reproduction service of the Bio-SAFE platform software to perform the actual reproduction. In both sexual and asexual reproduction, the parent or parents give some of their energy units to the child. Note that energy is conserved during reproduction, i.e. no energy is created as a result of reproduction.

Receive Message: This behavior is invoked by receive event behavior when a message for the Mobile-entity arrives. The Receive message behavior determines the message type and then invokes the appropriate behavior (e.g. relationships or service, described below) to process the message.

Send Message: This behavior is invoked by other Mobile-entity behaviors to send a message to a user or other Mobile-entity.

Relationships: This behavior enables a Mobile-entity to establish and maintain a limited number of relationships with other mobile-entities. Mobile-entities establish relationships under various circumstances. When a Mobile-entity detects another Mobile-entity on its Bio-SAFE platform, it can send a message requesting the establishment of a relationship to that Mobile-entity. Two mobile-entities can be introduced to each other by a third Mobile-entity, and parent mobile-entities establish relationships with their child Mobile-entity when it is born. Once a relationship is established between two mobile-entities, periodic relationship maintenance messages are exchanged. These messages contain information about the sender, such as its current location, energy level, age, and number of relationships it has. Mobile-entities use their relationships for a variety of purposes. For example, a Mobile-entity can detect the death of another Mobile-entity when it fails to respond to a certain number of relationship maintenance messages. A Mobile-entity can also gain information about the mobile-entities that it has relationships with because the relationship maintenance messages contain information about the sender. Consistent with the principle of local interactions described in Section 2.3, mobile-entities have a limited number of relationships and therefore can only interact with a limited number of other mobile-entities.

Service: This behavior enables a Mobile-entity to perform a service for human users or other mobile-entities. The recipient of the service must pay the Mobile-entity that performed the service with energy units. One possible service behavior is accepting requests for web pages and then delivering the web pages using the HTTP protocol. This service behavior is used by Aphid mobile-entities, described in Section 2.5.

Death: In Section 2.4.1, we stated that the resource control function of the Bio-SAFE platform kills mobile-entities when their energy units are exhausted. However, the death behavior enables a Mobile-entity to determine whether it should die before it exhausts its energy units.

.

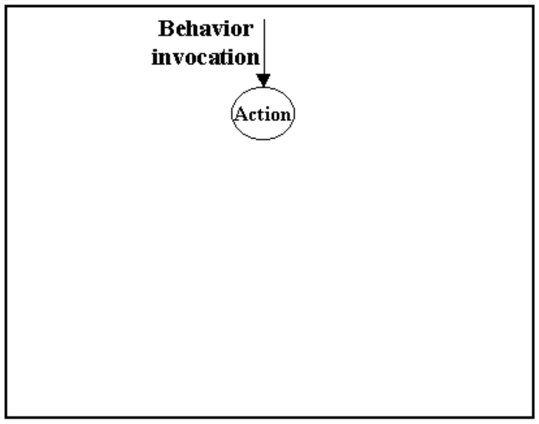


Figure 9a: Simple tree structure

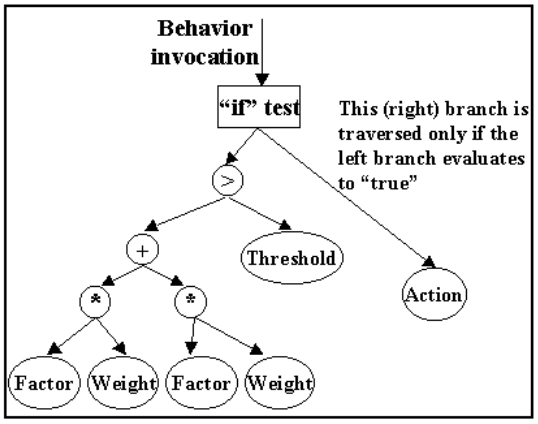

Behavior invocation
"if" test
This (right) branch is traversed only if the left branch evaluates to "true"
>
+
Threshold
Action
*
*
Factor
Weight
Factor
Weight

Figure 9b: More complex tree structure

For example, a Mobile-entity may choose to die because of old age or lack of use by human users.

Having described some possible Mobile-entity behaviors, we now describe the internal structure of Mobile-entity behaviors. Mobile-entity behaviors may contain blocks of executable code, numerical values, arithmetic operators, and control flow operators arranged in a tree structure. This tree structure is described by Koza in [11]. When a behavior is invoked, the tree is evaluated in depth first order. A tree structure allows executable code, numerical values, arithmetic operators, and control flow operators to be arranged in an infinite variety of ways to implement arbitrary functions. One simple arrangement is shown in Figure 9a. The node labeled "Action" is a block of executable code. Behaviors implemented with the arrangement shown in Figure 7 will execute its action code block whenever it is invoked.

A more complex arrangement is shown in Figure 9b. In Figure 9b, the nodes labeled "Factor" are blocks of executable code that return a numerical value based on local information or local interactions. Nodes labeled "Weight" are numerical values. The weight nodes derive their name from the fact that their values are multiplied with the return values of their associated factors. The node labeled "Threshold" is also a numerical value. A behavior implemented with the arrangement shown in Figure 9b will execute its action code block only if the sum of the products of the weighs and factors are greater than the threshold.

Implementing Mobile-entity behaviors using tree structures allows them to be easily modified in two ways. First, designers can modify a Mobile-entity behavior by adding factors, removing factors, or by changing the values of the weights and threshold. Second, the platform software can perform (automatic) crossover and mutation when mobile-entities reproduce. Crossover is implemented by grafting the tree structures of the parents' behaviors onto each other at random points to form the child's behaviors. Mutation is implemented by copying the weights and thresholds from the parents' behaviors with random changes to the child's behaviors. Since Mobile-entity behaviors are easily modified (by humans and automatically by the platform software), it is possible for applications to consist of mobile-entities with different or diverse behaviors. We believe that this diversity of behaviors, coupled with the natural selection process in the Bio-SAFE Architecture, will cause Mobile-entity behaviors to become more useful and/or efficient through evolution.

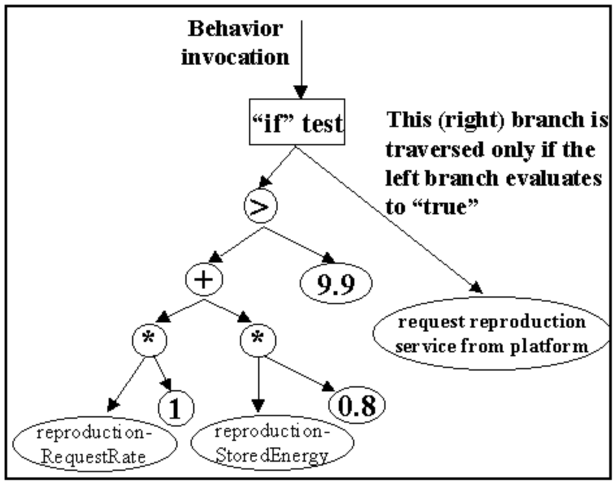

Behavior invocation
"if" test
This (right) branch is traversed only if the left branch evaluates to "true"
>
+
9.9
*
*
1
0.8
request reproduction service from platform
reproduction-RequestRate
reproduction-StoredEnergy

Figure 10: Example reproduction behavior

Factor Name/ Threshold	Weight/ Threshold Value
Reproduction Request Rate	1.0
Reproduction Stored Energy	0.8
Reproduction Threshold	9.9

Table 1: Factors, weights, and thresholds of example reproduction

We conclude this subsection with an extended example of a Mobile-entity behavior. Figure 10 depicts a possible mobile-entity reproduction behavior and its tree structure. The behavior consists of a *reproduction request rate* factor with a weight of 1, a *reproduction stored energy* factor with a weight of 0.8, and a reproduction threshold of 9.9. The factors, weights, and threshold of the example reproduction behavior are summarized in
Table 1. Even though the tree structure in Figure 10 more fully describes a behavior, we will use the table format in the remainder of the section because it is more compact. When the reproduction behavior is invoked, the *reproduction request rate* factor is first evaluated (recall the tree is evaluated in depth first order). The *reproduction request rate* factor returns a value from 0 to 20 based on the Mobile-entity's request rate. The return value of this factor is multiplied by its weight (1). The *reproduction stored energy* factor, which returns a value between –100 and 0 based on the Mobile-entity's energy level[4], is then evaluated. The return value of this factor is multiplied by its weight (0.8). The products are summed, and if the sum is greater than the reproduction threshold (9.9), the reproduction behavior will request the reproduction service from the platform software. Because the *reproduction request rate* factor

[4] The return values of the factors and the values of the weights and threshold were arrived at through ad-hoc calculations and experimentation. After gaining more experience with the Bio-SAFE Architecture, we hope to provide basic guidelines for the design of mobile-entity behaviors.

returns positive values, this factor encourages the reproduction behavior. Conversely, because the *reproduction stored energy* factor returns negative values, it inhibits the reproduction behavior. A factor may also return both positive and negative values.

2.4.2.3 Body

The body of the Mobile-entity will contain data. The data stored in the body of a Mobile-entity is related to the service behavior of that Mobile-entity. For example, if the service behavior of a Mobile-entity is to serve web pages, its body will contain web pages. When a Mobile-entity reproduces, the offspring is an exact copy of the parent's body. When two mobile-entities reproduce, the offspring may contain an exact copy of either parent's bodies or both parents' bodies.

2.5 Related Work

Researchers in the fields of Artificial Life [2][13][15] and Complexity [9][20] have studied large-scale biological systems and the behaviors of simple entities within those systems. That work has been concentrated in imitating life in a computer and understanding the basic processes. The Bio-SAFE Architecture applies the findings of those researchers in a new domain: the design and implementation of scalable, adaptive, and survivable/available network applications.

This project will not be the first to use biological principles in the design of network applications. [3] [10] used the immune system as a model for network security and intrusion detection. [1] [6] draws inspiration from the biological world in the design of scalable sensor networks. While those works parallel this project, the Bio-SAFE Architecture addresses a broader range of applications. It is also more open because the biological middleware (Bio-SAFE platforms and mobile-entities) allow other researchers to design and implement novel applications.

While there are similarities between the Bio-SAFE Architecture and active networks [18], they are different because they operate at different levels in the protocol stack. In active networks, executable code is injected into the network to modify behaviors of network elements at or below the transport layer. In the Bio-SAFE Architecture, mobile-entities containing executable code are also introduced into the network; however, mobile-entities act at the application layer and do not modify network layer behaviors.

Although the Bio-SAFE Architecture will make use of mobile agent middleware, it is different from the current mobile agent middleware systems [12] [14] [19] because it includes a paradigm based on biological principles and mechanisms. Network applications that conform to the paradigm of the Bio-SAFE Architecture will be able to scale, adapt to dynamic user demands and network conditions, survive failures, and remain available.

2.6 Future Work

This part describes our concept for applying biological principles and mechanisms to the design and implementation of distributed network applications. Although we do not have a complete understanding of the power and limitations of this biologically inspired approach, we believe that the biological extension to the previous research will prove to be a fertile area for many applications both military and non-military. It is proposed that large-scale biological systems should be studied, paying particular attention to how differentiated entities self-organize to form groups and/or hierarchies. We believe there are principles and mechanisms at work in this process that has relevance for the design of network applications. One needs to experiment, analyze and refine existing Mobile-entity behaviors by simulating conditions under different scenarios. In so doing one should design and document additional Mobile-entity behaviors that allow mobile-entities to create a rich web of relationships that span the global network. This web of Mobile-entity relationships can used to create a completely decentralized Mobile-entity directory system.

PART III: Development of a P System with Active Membranes for Application in a Bio Computing and Information System.

3.1 Introduction

The P system is a general distributed model, highly parallel, based on the notion of a **Membrane Structure**. Such a structure consists of several Cell-like membranes recurrently placed inside a unique “Skin” membrane. In the regions delimited by the membranes there are placed **Objects.** The objects can be transformed into other objects, can pass through a membrane or can dissolve the membrane in which it is placed. The P systems could be of different types, such as with Labeled membranes, Polarized membranes or Active membranes.

In the P System (21,22), we start with a certain number of objects in certain membranes and let the system evolve; if it will halt (no object can further evolve), then the computation is finished, with the result given as the number of objects in a specified membrane. If the development of the system goes for ever, then the computation fails to have an output. The system utilizes a **Priority** relation between evolution rules.
To enhance the parallelism of the P system, splicing of the membrane can be considered. The technique was originally presented by T. Head (23). This approach is very useful from the point of view of computational complexity. For example the **Satisfiability problem for propositional formulas** (in short the **SAT problem**) can be solved in linear time in such a frame-work.

3.2 The P Systems

The P system described above can be of different forms, such as:

1) P systems with Labeled Membranes
2) P systems with Polarized Membranes
3) P Systems with Active Membrane

A brief description of the different types of P systems mentioned above is given below:

3.2.1 P Systems with Labeled Membranes

The basis of this model is the fact that the parts of a biological system are well delimited by various types of “membranes”, in the broad sense of the term, starting from the membranes which delimit the various intra-cell components, going to the cell membrane, and then to the skin of organisms, and ending with more or less virtual “membranes” which delimit, for instance, parts of an eco-system.

3.2.2 P Systems with polarized Membranes

This P System uses a more realistic approach compared to the P System described in the previous section. In real cells, the molecules can pass through membranes mainly because of concentration difference in neighboring regions, or by means of electrical charges (ions can be transported in spaces of opposite polarization). This last variant is a much more restricted possibility as compared with the specification of the target membrane by its label: we only have two labels, + and -, associated in a non-injective way with the membranes. However, we know from the point of view of computational completeness that systems with three membranes suffice. Three membranes means a skin membrane and two inner membranes which can be labeled with + and -. Consequently using only polarized membranes we can still obtain computational completeness

3.2.3 P Systems with Active Membranes

In the P Systems described in the previous sections, it is assumed that the number of membranes can only decrease during a computation, by dissolving membranes as a result of applying evolution rules to the objects present in the system.

A natural possibility is to let the number of membranes also to increase during a computation, for instance, by division, as it is well known in biology. Actually from bio-chemical point of view, the membranes are not at all passive, like the membranes considered in the models discussed earlier. For example, the passing of a chemical compound through a membrane is often done by a direct interaction with the membrane itself (with the so-called **protein channels** or **protein gates** present in the membrane); during this interaction, the chemical compound, which passes through the membrane, can be modified while the membrane itself can in this way be modified (at least locally).

In P Systems with Active Membranes (24,25) the observation mentioned above, is used and the membranes play the central role in the computation. In this type of system, the evolution rules are associated both with the objects and the membranes, while the communication through the membranes is performed with the direct participation of the membranes; moreover the membranes can not only be dissolved, but they also can multiply by **division**. An elementary membrane can be divided by means of an interaction with an object from that membrane. Each membrane is supposed to have an “electrical polarization”, one of the three possible: **positive**, **negative**, or **neutral**. If, in a membrane, we have two immediately lower membranes of opposite polarizations, one **positive** and one **negative**, then that membrane can also divide in such a way that the two membranes of opposite charge are separated; all membranes of neutral charge and all objects are duplicated and a copy of each of them is introduced in each of the two new membranes. The skin is never divided.

In this way, the number of membranes can grow, even exponentially. As expected, by making use of this increased parallelism we can compute faster .For example, the SAT problem can be solved in this framework in linear time.

3.3 Future Work

The future research work on P system with Active Membranes should address the following:
1) Develop the scientific and mathematical basis of the model.
2) Develop the algorithms and the codes for the model.
3) Identify the Bio-Chemical experiments necessary for the implementation of the developed model in practice.

Conclusions

In conclusion the research has shown that the viability of a Bio Computing and Information System depends upon further research into the following areas:

1) Molecular Computation using Biological Cells: A Hybrid Approach .

2) The Bio-Safe Architecture: A Biologically inspired approach to the design of secure, scalable, adaptive and survivable Distributed Network Applications.

3) Development of a P- System with Active Membranes.

Therefore it is recommended that research be pursued in those areas.

References

1) Amorphous Computing web page. http://www.swiss.ai.mit.edu/projects/amorphous

2) Collins, R. and Jeffereson, D. AntFarm: Towards Simulated Evolution. In *Proceedings of Artificial Life II*, 1992

3) D'haeseller, P., Forrest, S., and Helman, P. An Immunological Approach to Change Detection. In *Proceedings of IEEE Symposium on Security and Privacy*, 1996

4) Dawkins, R. *The Selfish Gene*. Oxford University Press, 1989

5) Dumpert, K. *The Social Biology of Ants*. Pitman Publishing Inc., 1978

6) Estrin, D., Govindan, R., Heidemann, J., and Kumar, S. Scalable Coordination in Sensor Networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking* (MobiCom), 1999

7) Franks, N. Army Ants: A Collective Intelligence. *American Scientist*, 1989

8) Holland, J. *Hidden Order: How Adaptation Builds Complexity*. Addison-Wesley, 1995

9) Kauffman, S. *At Home in the Universe*. Oxford University Press, 1995

10) Kephart, J. A Biologically Inspired Immune System for Computers. In *Proceedings of Artificial Life IV*, 1994

11) Koza, J. *Genetic Programming II: Automatic Discovery of Resuable Programs*. MIT Press, 1994

12) Lange, D. and Oshima, M. *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley,

13) Millonas, M. Swarms, Phase Transitions, and Collective Intelligence. In *Proceedings of Artificial Life III*, 1994

14) Odyssey web page. http://www.genmagic.com/technology/odyssey.html

15) Ray, T. and Hart, J. Evolution of Differentiated Multi-Threaded Digial Organisms. In *Proceedings of Artifical Life VI*, 1998

16) Seeley, T. The Honey Bee as a SuperOrganism. *American Scientist*, 1989

17) Seeley *Wisdom of the Hive*. Harvard University Press, 1995

18) Tennenhouse, D., Smith, J., Sincoskie, W., Wetherall, D. and Minden, G. A Survey of Active Networks Research. *IEEE Communications Magazine*, January 1997

19) Voyager web page. http://www.objectspace.com/products/voyager.

20) Waldrop, M. *Complexity.* Simon & Schuster, 1992

21) Gh. Paun, “Computing with membranes”, J. Computer System Sciences, 61 (2000), and TUCS Research Report No. 208, November, 1998.

22) C.S. Calude, Gh. Paun, “Computing with cells and atoms: An introduction to Quantum, DNA and Membrane computing”, Taylor and Francis Inc. 2001.

23) T. Head, "Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors", Bulletin of Mathematical Biology, 49 (1987), 737-759.

24) Gh. Paun, "P systems with active membranes: Attacking NP complete problems", J. Automata, Languages, Combinatorics, 5, 2000, and CDMTCS Research Report No 102, 1999.

25) Gh. Paun, "Membrane Computing – An Introduction" Springer: 2002.

Bibliography

1) Abelson, Harold, Don Allen, Daniel Coore, Chris Hanson, George Homsy, Thomas F. Knight, Radhika Nagpal, Erik Rauch, Gerald Jay Sussman and Ron Weiss. 2000. Amorphous computing. Communications of the ACM 43(5):74–82.

2) Adleman, Leonard M. 1994. Molecular computation of solutions to combinatorial problems. Science 266:1021–1024.

3) Coore, Daniel N. 1999. Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer. Ph.D thesis, MIT Department of Electrical Engineering and Computer Science. http://www.swiss.ai.mit.edu/projects/amorphous/papers/coore-phdthesis.ps

4) Elowitz, Michael B., and Stanislas Leibler. 2000. A synthetic oscillatory network of transcriptional regulators. Nature 403:335–338.

5) Gardner, Timothy S., Charles R. Cantor and James J. Collins. 2000. Construction of a genetic toggle switch in Escherichia coli. Nature 403:339–342.

6) Knight, Thomas F., Jr., and Gerald Jay Sussman. 1998. Cellular gate technology. In Unconventional Models of Computation, ed. C. S. Calude, J. Casti and M. J. Dinneen, pp. 257–272. Singapore, New York: Springer.

7) McAdams, Harley H., and Adam Arkin. 2000. Towards a circuit engineering discipline. Current Biology 10(8):R318–R320.

8) Monod, J., and F. Jacob. 1961. General conclusions: Teleonomic mechanisms in cellular metabolism, growth and differentiation. Cold Spring Harbor Symposia on Quantitative Biology: Cellular Regulatory Mechanisms 26:389–401.

9) Pérez-Rueda, E. and Collago-Vides, J. 2000. The repertoire of DNA-binding transcriptional regulators in *Escherichia coli* K-12. Nucleic Acids Research 28, 1838-1847.

10) Rössler, Otto E. 1974. Chemical automata in homogeneous and reaction-diffusion kinetics. In Physics and Mathematics of the Nervous System, ed. M. Conrad, W. Güttinger and M. Dal Cin, pp. 399–418. Berlin: Springer-Verlag.

11) Weiss, Ron. 1999. Programming colonies of biological cells. Ph.D. thesis proposal, Massachusetts Institute of Technology. http://www.swiss.ai.mit.edu/~rweiss/ bio-programming/phd-proposal.pdf

12) Weiss, Ron, George E. Homsy and Thomas F. Knight, Jr. 1999. Toward in-vivo digital circuits. DIMACS Workshop on Evolution as Computation.
http://www.swiss.ai.mit.edu/ ~rweiss/bio-programming/dimacs99-evocomp.ps

13) M .P .Robertson, J. Hesselberth, J.C. Cox & A. D. , Ellington, " Designing and selecting components for Nucleic Acid Computers". DIMACS series in Discrete Mathematics and Theoretical Computer Science, Vol.54, 2000.

14) A.J. Turberfield, B. Yurke & A. P. Mills,Jr. " DNA Hybridization Catalysts and Molecular Tweezers". DIMACS series in Discrete Mathematics and Theoretical Computer Science, Vol.54, 2000.

15) M.A. Nielson & I. L. Chuang, " Quantum Computation and Quantum Information". Cambridge University Press, 2000.