

White Paper: Engineering Emergent Protocols

Stephen F. Bush
Amit B. Kulkarni

ABSTRACT

Two approaches are proposed for revolutionary gains in MEMS device communication. The first is to view all network devices as computational or active devices. Computation can take many forms. The amount of computation may vary, but every device has some type of computation, either programmed or ambient. Use of computation in an optimal manner is the same challenge faced by active networks. Thus, advances in active networks and networks of MEMS devices are mutually beneficial. The second approach is to optimize networks of MEMS devices via exploiting emergence. Understanding emergence requires understanding complexity; this relationship is explored relative to networking. The results lead toward emergence that can be precisely engineered to achieve desired characteristics.

The purpose of the research proposed in this white paper is to understand and develop a theory for basic emergent protocols using computer science formalisms and simulation techniques for higher order emergent protocols. Emergence is a concept or property in the macroscopic behavior of a system that is not contained within the microscopic description (Figure 1). It is a natural phenomenon in which structured collective behavior appears from the interaction of simple subsystems, such as MEMS devices. The research proposed in this white paper will integrate emergent protocols in network service protocols to create a new class of featherweight, survivable protocols called “emergent protocols.” The challenge will be to identify emergent protocols that arise from the operation of the device and its interaction with other MEMS devices, especially in an environment of suboptimal connectivity where messages can be lost and the system must operate with only a partial knowledge of its environment.

By understanding the emergence of new behaviors in such a dynamic environment, the proposed research will induce emergent protocols that imple-

ment required network service protocols. A key is to recreate and study the behavior of large multitudes of interacting MEMS devices.

Emergence is a natural phenomenon in which structured collective behavior appears from the interaction of simple subsystems, such as MEMS devices. The research proposed in this white paper will integrate emergent behavior in network service protocols to create a new class of featherweight, survivable protocols called “emergent protocols.”

Another key is to integrate as much communication and processing into the devices’ normal operating physical behavior as possible and minimize or eliminate the need for a separate, full-size, higher power processor external to the system to set up, manage, or control the system. Existing computer communication protocols, while supporting the explosion of today’s Internet, are a poor fit on MEMS devices. This is primarily because communication protocols have been designed assuming the devices supporting them will be modified to support the protocol. Such modification assumes the supporting devices have the appropriate processing capability, are generally static in movement, and exhibit limited dynamics and physical interaction.

In active networks, benefits and insights have been gained by removing the taboo of the simple “network versus complex end-system” myth. In this project, the myth of protocols will be removed entirely, viewing communication networking from a clean slate.

Two general approaches are proposed for improving MEMS device communication:

- View all network devices as computational or active devices. The computation can take many

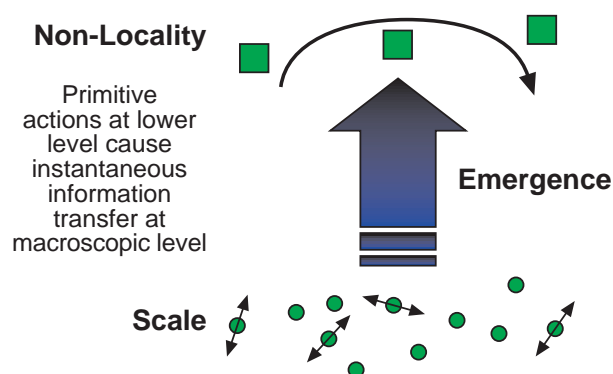


Figure 1. Microscopic versus macroscopic scales.

forms. The amount of computation may vary, but every device has some type of computation, either programmed or ambient. Use of computation in an optimal manner is the same challenge faced by the active network community. Thus, advances in one area benefit the other.

- The solution to optimizing active networks and networks of MEMS devices can be found in exploiting emergence. Understanding emergence requires understanding complexity; this relationship is explored relative to networking.

The MEMS environment is antithetical to the existing Internet legacy in every respect. Implementing standard communication protocols in such an environment is not feasible; they add significant overhead that can lead to performance degradation. However, by viewing the problem in a revolutionary manner, immense gains in communication and processing can be achieved. For example, the universe is a computational engine whose power has only been infinitesimally exploited. Accurate predictions of phenomena ranging from the motion of planetary bodies to the behavior of fluids are readily available. These phenomena are mathematical functions. The initial physical state is the input to the function, the physical actions are the computation, and the final states of the physical entities are the output of the function. MEMS devices reside in this universe of computation, known as “ambient processing.” The billiard ball computer is an easy, clearcut example of ambient processing that uses nothing more than the physical properties of balls and reflectors to perform computation.¹³ Further, as part of the physical universe, MEMS devices also conform to well known, predictable physical behavior that is termed “intrinsic processing” (Figure 2).

The fine-grained fusion of physics and information required to achieve an Internet of MEMS devices will be enabled by ambient and intrinsic processing in conjunction with the physics of information.

Information also exhibits predictable phenomena with regard to processing and transmission that is analogous to the classical physics of matter and energy.²⁶ This is referred to as the “physics of information.” Information has randomness, entropy, and even thermodynamic properties. Computation in support of communication also has predictable, bounded behavior in terms of the smallest executable program capable of representing information, known as “Kolmogorov Complexity.”²⁰ Immense

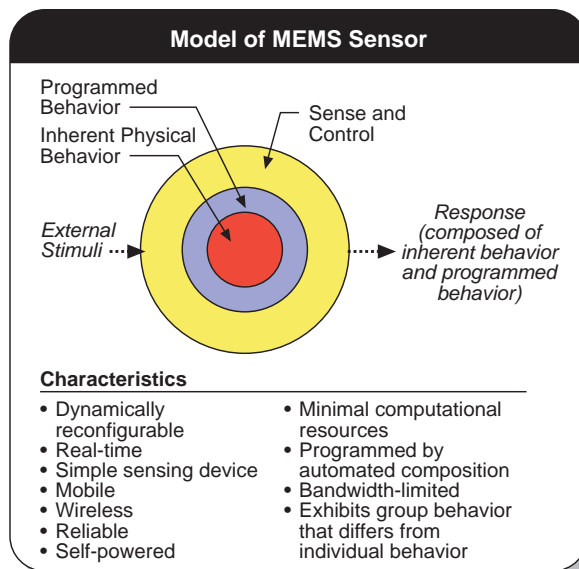


Figure 2. Viewing a MEMS device as an active network component.

gains in communication and processing can only be achieved if the physics of information and the physics of matter and energy are studied in concert and their synergies exploited. The goal is to minimize the processing required for communication in MEMS devices by obtaining synergy among ambient processing, intrinsic processing, and the physics of the information to be transmitted. The fine-grained fusion of physics and information required to achieve an Internet of MEMS devices will be enabled by ambient and intrinsic processing in conjunction with the physics of information.

The goal is to minimize the processing required for communication in MEMS devices by obtaining synergy among ambient processing, intrinsic processing, and the physics of the information to be transmitted.

The challenges in the research proposed are to understand the physical behavior of devices and the global impact when swarms of such devices are present in a system. Exhibited global system behavior (also called “emergent protocols”) may be radically different from individual device behavior because each device gathers information about its immediate environment, processes the information, and reacts to it in isolation. In current distributed systems, global behavior is induced by programming each component of the system and by attempting to control system behavior using feedback control techniques. In essence, the system is forced into a desired behavior instead of synergistically exploiting system behavior. In the process,

the system is overprogrammed. This approach is unsuitable for large numbers of low-power components that are intimately influenced by their environment.

A simple illustration of emergent protocols would be a typical Internet routing protocol in a highly dynamic MEMS environment—Smart Dust,²² for example. The MEMS devices, sensors, and actuators the size of dust particles floating in the air would move so dynamically in three dimensions that the routing algorithm would never stabilize, as illustrated in the top of Figure 3.

An algorithm maximizing the use of ambient processing, intrinsic processing, and the physics of the information in an emergent manner is illustrated in the lower part of Figure 3. Current routing algorithms generally assume fixed address hierarchies stored in intermediate forwarding devices.

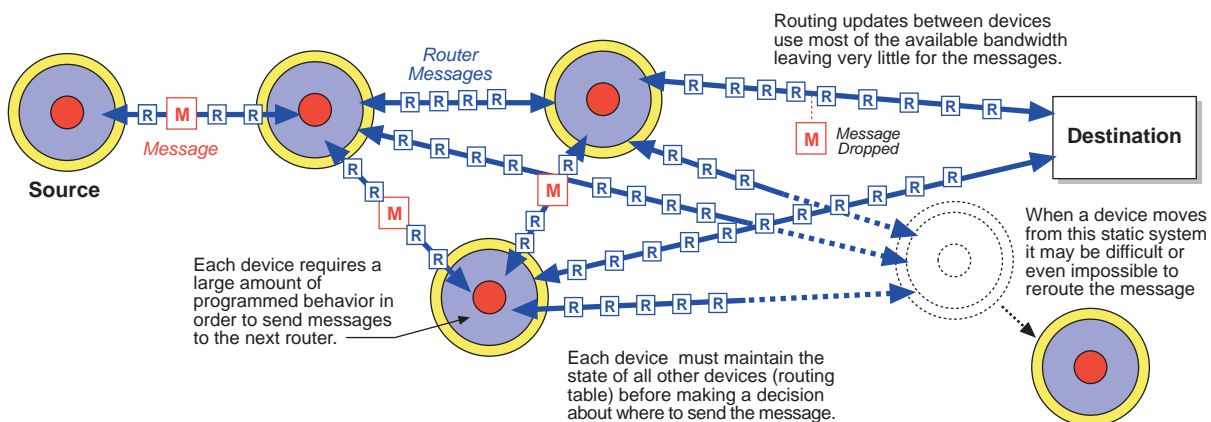
Storing and maintaining this database of routing information is resource intensive and clearly inefficient. Instead, the routing should be integrated with the environment and routing based upon gradients from sensor readings, as an example. While the ability to encode knowledge of the physics of the environment may appear to be irrelevant to communications, it will result in a more efficient and integrated system of devices with its environment.

HARNESSING EMERGENT PROTOCOLS

Emergent behavior already present in the system will be harnessed to develop ultralow overhead network services.

Work in the study of emergence is ongoing within many disciplines, ranging from economics and soci-

Routing Example Without Emergent Behavior



Routing Example With MEMS Sensors and Emergent Behavior

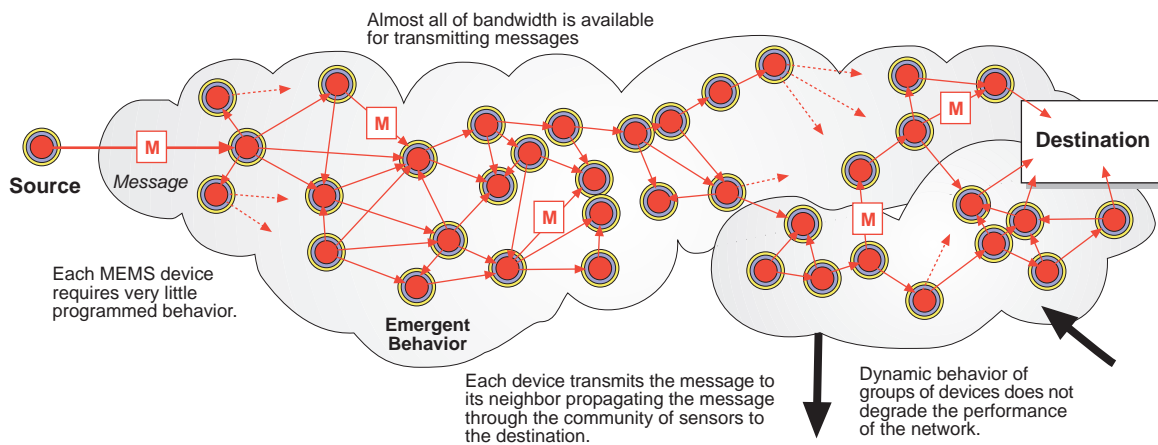


Figure 3. Better routing through emergence. Top: MEMS devices, sensors, and actuators the size of dust particles floating in the air would move so dynamically in three dimensions that the routing algorithm would stabilize. Bottom: An algorithm maximizing the use of ambient processing, intrinsic processing, and the physics information in an emergent manner.

ology to biology, physics, and computer science. The growing-point language may be viewed as emergence utilized for amorphous computing.⁸ The computational particles in the amorphous medium all have the same program and are aware only of their neighbors, yet they can be coaxed into forming patterns capable of performing Boolean logic. Progress in the study of emergence itself has been made.^{9,10,19} Crutchfield and Mitchell¹⁹ also applied new methods for detecting emergent computation in nonlinear processes using genetic algorithms. They focused on solutions for frameworks belonging to an idealized class in which the topology is fixed but the information processing itself is global.

The proposed effort will apply the results from this work within the scope of communications of many low-bandwidth and low-compute-power devices. By attempting to apply fundamental emergence results, this project will also find ways to better extend the fundamental research into emergence.

Employing emergence in a network context has been suggested by Fisher and Lipson.¹² While understanding that emergence can yield smaller, more efficiently programmed software and network layers, they appear to be more focused on the current Internet, security, and survivability rather than on MEMS device performance, where this proposed effort expects a high impact.

One can view emergence as a process that leads to the appearance of structure that is not directly described by the defining constraints and instantaneous forces that control a system. Thus, for example, new movement patterns can arise over time that are not directly specified by the equations of motion. As shown in Figure 4, every simple mobile sensor (belonging to a group of reconnaissance sensors) may be directed to follow a “self-avoiding random walk,” i.e., the sensor may choose to move in any direction except the direction it came from. While the behavior of individual sensors may appear to be chaotic, the group as a whole will end up tracing a fractal-like self-similar pattern. This emergent protocol can be harnessed to control system behavior while keeping the individual behaviors simple. Instead of programming every sensor to trace a particular route to perform the reconnaissance and then programming the system to handle every individual sensor position, coverage, interaction with other sensors, and other specifics, a set of simple commands can achieve the desired result if the emergent protocols is used advantageously.

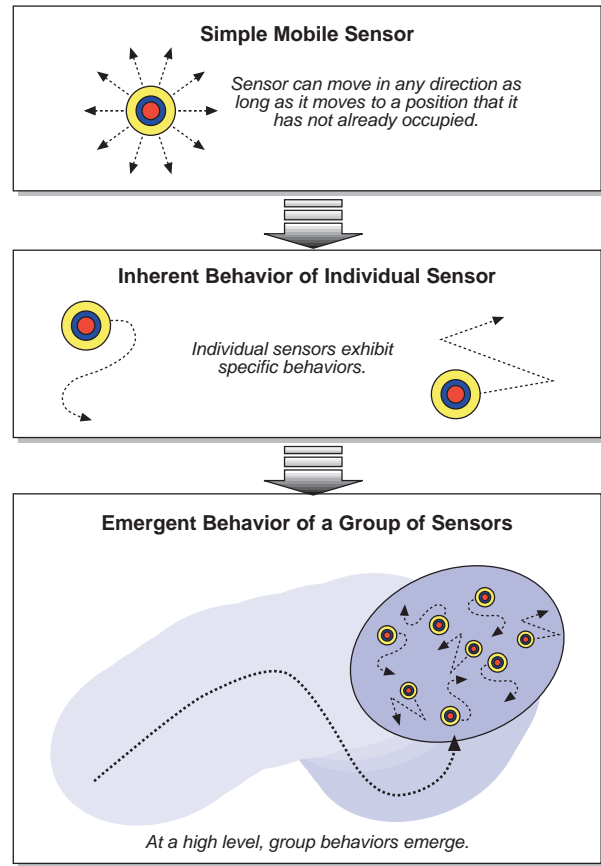


Figure 4. Example of emergent group behavior. while the behavior of individual sensors may appear to be chaotic, the group as a whole will end up tracing a fractal-like self-similar pattern.

CLASSIFYING EMERGENT PROTOCOLS

Studies on emergent protocols have shown certain characteristics of systems provoke certain types of behavior. One classic example is that of the Alife system, which simulates the behavior of a colony of ants foraging for food. Another example is that of a group of fireflies that synchronize on a common blinking period. These behaviors are influenced by an important underlying characteristic of the system: its information exchange technique. In this section, we classify various emergent protocols based on the type of information exchange taking place between the entities of the system. For each class, a sample simulation situation is discussed along with a potential application towards a solution for a networking problem.

Time synchronization

- *Messaging type:* State diffusion
- *Simulation:* Firefly
- *Description:* This type of information exchange takes place in a group of fireflies. Initially, a

group of flies flash with a random period. The flashes are timed by the progressive excitation of a chemical in each firefly. If a firefly senses a certain amount of luminescence from its neighbors, it resets its excitation in order to flash simultaneously with its neighbors. Thus the state of each firefly is diffused into its environment, which allows others to react to the received state. After some time, all flies blink in unison. A variance on this theme could be a situation in which the environment alters the state as it is being diffused.

- *Potential application:* Synchronization for contention avoidance in networking protocols.

Random neighbor

- *Messaging type:* Direct synchronous
- *Simulation:* Mousetraps
- *Description:* A group of ready-to-spring mousetraps are strewn randomly. One of the mousetraps is triggered, which in turn triggers one or more randomly selected neighboring mousetraps and itself becomes dormant. The triggered mousetraps trigger others and this process continues indefinitely. This simulation models rapid information transfer through a group.
- *Potential application:* This type of scenario is already utilized in networking in what are called ‘gossip’ protocols, in which state information is sent to a random neighbor.

Scent marking

- *Messaging type:* Indirect soft-state
- *Simulation:* Bees
- *Description:* Each bee in a random collection moves randomly and drops a trace chemical behind itself as it moves. The amount of chemical present in the bee’s current path determines how far the bee turns and moves. This behavior leads to the creation of honeycomb structures.
- *Potential application:* Automatic domain establishment for dynamic coalition.

Predator-prey

- *Messaging type:* Direct resource consumption
- *Simulation:* Rabbits and grass ecosystem
- *Description:* Rabbits move randomly and grass grows randomly as well. If a rabbit encounters grass, it eats the grass and gains energy. Upon gaining enough energy, the rabbit reproduces. If its energy drops below a certain level, it dies. Thus, this simulation models an ecosystem with competing entities. It can be modified to create entities that evolve more complex features to gain an advantage.

- *Potential application:* Optimal resource allocation in competitive systems, self-evolving fault and attack tolerant systems.

Diffusion limited aggregation

- *Messaging type:* Diffusion and aggregation
- *Simulation:* Fractals
- *Description:* Particles move randomly in the vicinity of a cluster until they collide with it and become part of it. The resulting formation is a dendritic fractal.
- *Potential application:* Self-organizing symmetric coalition

RELATIONSHIP BETWEEN EMERGENCE AND COMPLEXITY

Emergence and complexity are intimately related. Emergence appears as unexpected order in the midst of apparent randomness. Emergence is unlikely to be found in total order, or in total randomness, but rather somewhere between those extremes. Kolmogorov Complexity, discussed previously with regard to the physics of information, is one of many measures of complexity that can be used to test for the onset of emergence. Kolmogorov Complexity is a particularly attractive measure in an active network environment because it is the size of the smallest program capable of representing a given string. This measure is in support of the goal with resource-constrained MEMS devices to minimize programming, particularly programming in support of network services. Further details are described in the next section.

New tradeoffs between computation and communication have recently been explored in the arena of active networking.³ Research on active networks has shown that injecting some amount of application-specific intelligence in the network improves application performance, especially in wireless networks.¹⁶ Active networks also offer decentralized computation through the use of active packets that carry code, and they offer global coordination through the use of a transient “soft state” that can be created and shared by the active packets in an active network. Active networks offer the potential to develop and implement algorithms that can detect and recognize global emergent protocols and act on it.

A salient question that needs to be answered while programming an active network application or service is, “How much computation should be injected in the network?” The network may or may not have enough processing power, bandwidth, or both. Injecting an extra layer of processing can defeat the usefulness of the code. The best case sce-

nario unfolds when the injected code stimulates behavior in synchrony with the behavior of the system, so performance is enhanced. GE Corporate Research and Development is already studying the use of complexity theory to evaluate criteria for developing programs that will be carried in active packets.^{3, 17} The proposed research will be extended to take into consideration the randomness of system behavior and the presence of emergent phenomena. The goal of this proposed effort is to extend this work to nonidealized classes of frameworks in which global information processing is required on systems whose topology can change dynamically. In particular, the goal will be to determine how to reduce network service overhead by identifying and harnessing emergent protocols.

The best case scenario unfolds when the injected code stimulates behavior in synchrony with the behavior of the system, thus enhancing performance.

Emergence and complexity are deeply intertwined. Thus, a deeper understanding and new methods of quantifying and discovering underlying properties are necessary for optimal utilization of an active network. This leads to better understanding of the new dimension enabled by active networking, which is determining when and how much information should be active versus passive to obtain optimal performance from an active network.

THE SHANNON AND KOLMOGOROV RELATIONSHIP IN ACTIVE NETWORKS

Kolmogorov Complexity is a measure of descriptive complexity contained in an object.¹¹ It refers to the minimum length of a program such that a universal computer can generate a specific sequence. Kolmogorov Complexity is related to Shannon entropy, in that the expected value of $K_\phi(x)$ for a random sequence is approximately the entropy of the source distribution for the process generating the sequence. However, Kolmogorov Complexity differs from entropy in that it relates to the specific string being considered rather than the source distribution. Kolmogorov Complexity can be described as follows, where ϕ represents a universal computer, p represents a program, and x represents a string:

$$K_\phi(x) = \left\{ \min_{\phi(p) = x} l(p) \right\}$$

Random strings have high Kolmogorov Complexity on the order of their length, because pat-

terns cannot be discerned to reduce the size of a program generating such a string. On the other hand, strings with a large amount of structure have fairly low complexity. Universal computers can be equated through programs of constant length; thus a mapping can be made between universal computers of different types, and the Kolmogorov Complexity of a given string on two computers differs by known or determinable constants. The Kolmogorov Complexity $K_\phi(y|x)$ of a string y given string x as input is described by the following equation:

$$K_\phi(y|x) = \left\{ \begin{array}{l} \min_{\phi(p, x) = y} l(p) \\ \infty, \text{ if there is no } p \text{ such that } \phi(p, x) = y \end{array} \right\}$$

where $l(p)$ represents program length p and ϕ is a particular universal computer under consideration. Thus, knowledge or input of a string x may reduce the complexity or program size necessary to produce a new string y . A critical difficulty with Kolmogorov Complexity is that it cannot, in general, be absolutely computed. Any program that produces a given string is upper-bound on the Kolmogorov Complexity for this string. Recent advances in estimation of Kolmogorov Complexity have been developed.^{7, 25}

Shannon entropy is compared and contrasted with Kolmogorov Complexity in networking in Figure 5. In a nonactive legacy network, the only option available is to transmit bits whose interpretation at an end-system is predetermined. The bits in a legacy network can be compressed using Shannon encoding techniques, but without an understanding of Kolmogorov Complexity, it is not apparent

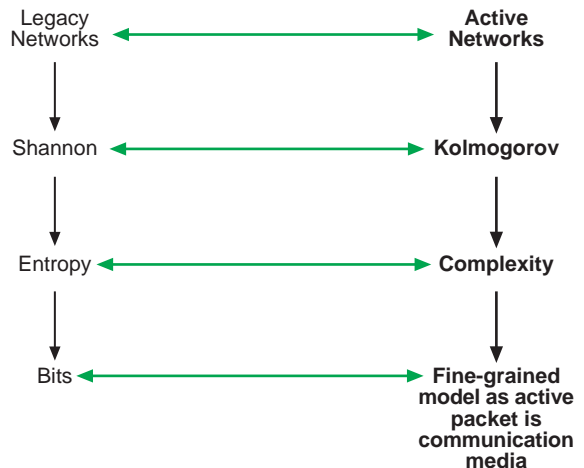


Figure 5. Old versus new view of network fundamentals for MEMS communication.

whether those passive bits are providing the same impact per bit in supporting network performance. Clearly, the more complex the data in the Kolmogorov sense, the less gain there is in transmitting it in executable model form. Conversely, the simpler and larger the information to be transmitted, the more gain there is in transmitting it as an executable model. It is also possible that portions of the data may be best transmitted as model and portions as passive data. The optimal mix can obviously be determined by estimating the Kolmogorov Complexity. A detailed study of complexity, particularly Kolmogorov Complexity, is required.

Ongoing study: complexity for active networking

The primary goal of the ongoing work has been the study of the fundamentals of complexity for active networking. One purpose was to ascertain if estimates of Kolmogorov Complexity can be determined in real time accurately enough to be used for real-time network operation. This has been successfully demonstrated through use of complexity probes.

In addition, this study provides a framework for exploring the relationships between various active and passive components—that is, program versus data. The resulting framework developed in this study will be used to determine how complexity relates to fault detection and reconstitution for self-healing. The system used for experimental study of complexity should mirror a large-scale, highly dynamic yet easily controlled environment that is representative of actual information sensed and computed in a variety of real world information systems. The approach taken in this effort to understand the evolution of complexity is illustrated in Figure 6.

A Swarm model²⁷ has been developed that provides an open framework for experimentation on complexity theory and its relationship to active net-

working. The Swarm emulation has been programmed with three types of entities:

- Turing Machines
- Programs
- Bit-strings

Multiple entities of each type exist and are represented in Figure 6. The program (MEMS programmed behavior) is represented by a set of states connected by transitions, the bit-string data (MEMS sensed or communicated information) by binary digits within a rectangular array, and the Turing Machine (MEMS hardware) by the computer chip shown in Figure 6. Each entity is placed in a random location within a field of attractive force causing bit-strings to be attracted toward programs and programs toward Turing Machines.

The system can be viewed as a two-dimensional grid with a possibility of four types of objects residing on any grid location, as shown in Figure 6. All interactions and control decisions are made by the local entities; there is no global control.

The motivating factor for each of the entities is heat. Each entity generates a small amount of heat, determined probabilistically within a range specified as a startup parameter. Heat can represent the initial flow of information that causes one entity to become interested in working with another and form groups that are not too small or too large. Using energy to represent information is discussed further by Harmon.¹⁴ This clustering behavior is the result of the entities' desire to maintain an ideal temperature. They will cluster via a randomized movement pattern in a direction seeking to maintain this temperature.

Heat diffuses through the system in a realistic manner. The brightness indicates the estimated complexity of an entity: dark green indicates lower complexity, and bright green indicates higher complexity. To provide a preview of how this emulation is designed, Figure 7 shows the emulation at 100 and 300 time units, respectively. (This figure should be viewed in color for full interpretation.) If a Turing Machine, program, and data input tape meet, the program is executed and the program output tape generated by program execution is added to the system—available for use as data input in further computations. Figure 7 highlights two clusters. The circled cluster on the left is dark because the bit-strings have few Turing Machines to enable computation, while the right cluster is brighter because it contains more computational activity and growing complexity per bit-string. The clusters are attracted to one another and eventually form a single, larger, brighter cluster.

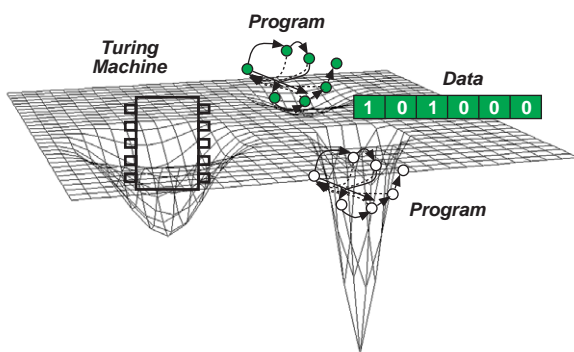


Figure 6. Emulation components and high-level dynamics.

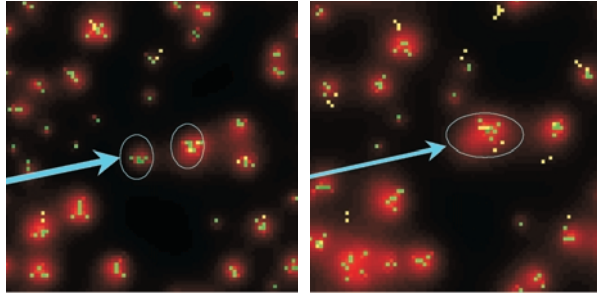


Figure 7. Cyberswarm simulation at 100 and 300 time units. Entities and their representation in the emulation are indicated below:

Entity	Representation
Turing Machine	Yellow—only when active
Tape	Green —lighter shading when more complex
Turing Machine program	Green —lighter shading when more complex
Heat	Red—darker when hotter

Two levels of abstraction occur simultaneously in this system:

- Individual, goal-directed behavior and information exchange
- Computation

In the first level, heat is a representation of motivation for movement toward common areas. The entities are initially located randomly throughout the two-dimensional space and gradually cluster, seeking to reach an ideal temperature. Complexity exists in the location and movement of the entities. Movement toward common areas allows information exchange to occur. At this level of abstraction, concepts such as access control in the exchange of information can be explored (beyond the scope of this paper). The second level is focused on compu-

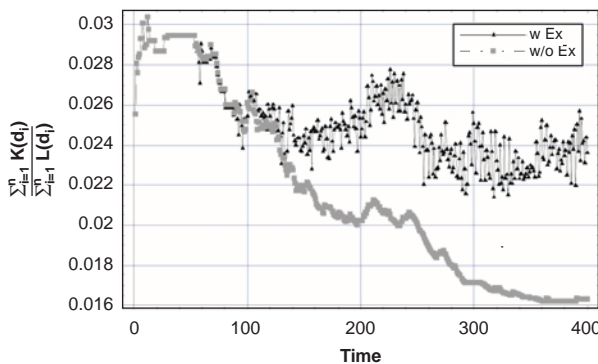


Figure 8. Complexity per length.

tation, the primary focus of this paper. The first level of abstraction provides the highly dynamic system in which the second level, computation, takes place.

In Figure 8, the complexity per length of data is plotted. With program execution enabled, the complexity per length of data increases. In the case of operation without program execution, only a finite length of data exists. When the total estimated complexity of the data within each entity is plotted over time, the system with program execution shows a marked increase in estimated complexity over that of the pure exchange system.

In Figure 9, the expected value of the complexity of all bit-strings contained by an entity is plotted versus time. The larger curve shows the average complexity per entity when program execution is enabled. As expected, program execution with this particular program, one that shuffles data, increased the average system complexity. Note that the difference in complexity does not appear until approximately time 100. However, program execution events began slightly before time 100. The delay between the time of the cause of complexity change and the time for a measurable increase in complexity to occur is important, but beyond the scope of this paper.

Emergence in networks

This emulation validated several intuitive concepts. The first involves complexity in a closed system, which has a fixed number of entities in which no data, programs, or Turing Machines may enter from outside the system. A program injected into the system that microscopically increases complexity generated greater average macroscopic system complexity. The results of this study experimentally validate the use of our complexity probes to detect complexity within a system. In addition, this study shows that groups of many simple devices can be coordinated to work with environmental condi-

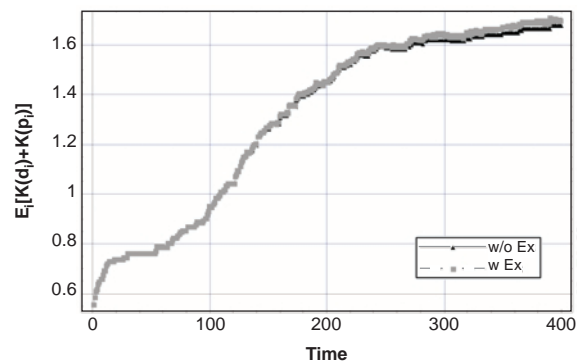


Figure 9. Expected complexity over time.

tions, such as heat. In addition, communication occurs not only by sensing environmental conditions and exploiting emergent behavior, but also by the exchange of executable programs, that is, the Turing Machines.

A complexity analysis tool should quickly and efficiently identify and display the complexity of an information system ranging from an application to a node, network, or an interconnection of networks. The tool should be portable, easy to use, have minimal impact upon the system, and be integrated with the management of the system or network. One approach has been to use lessons learned from the emulation in this analysis to consider the design of a complexity-based analysis system within the context of network management. The Swarm framework in the emulation previously described queries potentially large numbers of entities, such as Simple Network Management Protocol (SNMP) polls data for system management. A salient difference is that in a Swarm model, simulation time can be controlled so all data is queried at precisely the requested simulation step interval. However, design and minimization of the number of data points to be collected, such that system operation can be described as fully as possible, is exactly the same goal in management information base design for system and network management.

While integration with system management using a de facto standard such as SNMP is a future goal, a set of Java packages implementing lightweight “probes” that transparently gather data from a bit stream and report the result to Java’s vulnerability analysis portion has been developed. This design was chosen because the implemented probes are extremely lightweight and initiate the transfer of bit-level data, rather than waiting to respond to a management query. Once a better understanding of complexity is obtained, transition to implementation in SNMP will likely take place. The Swarm emulation ran with 200 such probes, one in each entity object, and successfully reported the results to the analysis package.

GE Corporate Research and Development intends to pursue the study of complexity within an active network environment,^{3,23} particularly with regard to network reconstitution in the face of attack. A more detailed analysis of the complexity emulation results is required, as well as modifications to the basic emulation presented here, to understand the role of complexity in automated service composition using MEMS emergent protocols.

GE Corporate Research and Development is already studying the use of complexity theory to evaluate criteria for developing programs that will be carried in active network packets. The research proposed in this white paper will be extended to take into consideration the randomness of system behavior and the presence of emergent phenomena.

EMERGENCE AND RESOURCE CONSUMPTION

Using the complexity probes, the onset of emergence can be detected in a system and controlled such that the desired emergent properties are maintained. Figure 10 shows a classical emergent pattern formed from eight independent interacting rules. The emergent pattern in Figure 10 is formed from a single row of cellular automata that evolve in the vertical direction of the figure. The light-colored squares are “on” and the dark are “off.” Figure 11 shows the density of the “on” squares (bits) as the pattern evolves.

In Figure 12 the frequency of rule activation is shown as the system evolves. Lighter areas indicate a higher frequency of activation. There are eight rules, and the system evolves through 50 transitions. The first rule is activated when neighboring cells are “off.” Thus it is activated quite often. There is a pattern to the frequency of rule activation. Each rule consumes a known amount of power. Thus it is desirable to design the emergent system such that rules that use more power are activated less frequently while achieving the same macroscopic behavior.

In Figure 13, the linear row of cellular automata is expanded with an additional dimension to a study

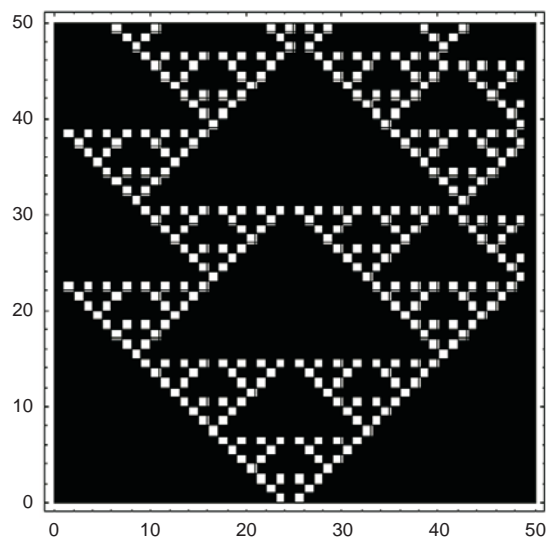


Figure 10. Emergent structure.

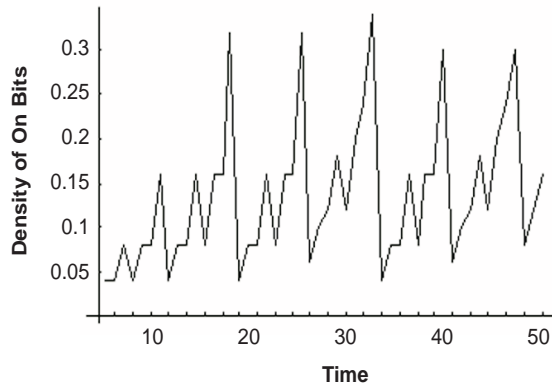


Figure 11. Density (power utilization) of “on” bits.

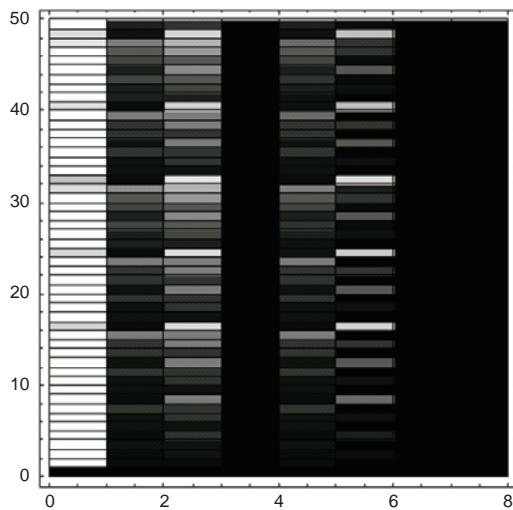


Figure 12. Frequency of rule activation.

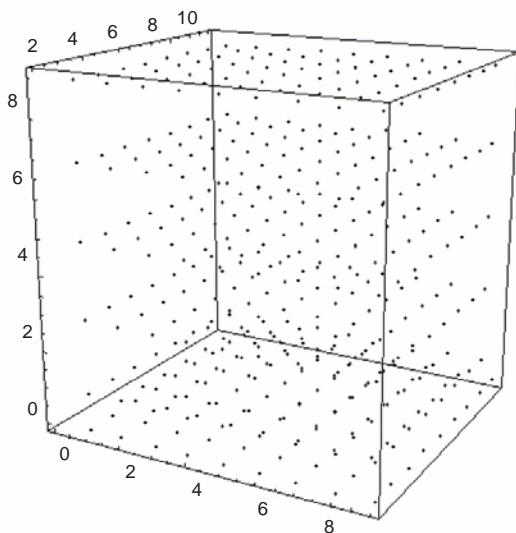


Figure 13. Evolution of two-dimensional emergence.

of emergence in the case of a two-dimensional grid. The evolution of the “on-off” pattern takes place in the vertical dimension.

An emergent mechanism requires communication within a given neighborhood of cells of size N_h . A global communication and control mechanism requires communication on the order of the size (number of nodes) of the network, N_s . While each of the N_h cells communicates concurrently, the emergent mechanism can take several iterations, T_i , to reach a desirable state. On the other hand, a global control mechanism can require several exchanges, E_x , to take place. These exchanges must occur within the entire network, N_s . E_x is proportional to the complexity of the control operation and the reliability built into the protocol, e.g., retransmission capability. The assumption is that resources are proportional to both the number of transmissions required as well as the distance between nodes.

Let D_n be the average distance between nodes within either N_h or N_s . In this simplified analysis, it is assumed that sensors can perform data-forwarding. However, the data-forwarding resources utilized are included in D_n . Consider the case where $T_i = E_x$ and D_n varies. Assume uniformly random scattering of nodes. A critical component is the difference in the number of interactions required in global versus local mechanism to reach the same desired state or objective. The graph in Figure 14 assumes that the same number of interactions or protocol exchanges are required in both cases. If so, the emergent mechanism is more efficient. However, if the rules require fewer interactions than the global mechanism, then the efficiency is even greater.

The resources consumed versus the locality, or number of nodes that are required to participate in a rule, are plotted in the graph shown in Figure 15. As long as locality can be kept low, the emergent system is more efficient.

Emergent synchronization can be applied to featherweight network services in various ways, eliminating collisions in transmission, topology discovery, and information broadcast. The following example, developed at the GE Research and Development Center, demonstrates some of the concepts being explored. Emergent protocols is induced and harnessed to improve network performance by synchronizing groups of devices transmitting on the same channel. Each device makes only local decisions regarding the transmit algorithm, but emergent properties of the system are exploited to reduce collisions. Figure 16 shows the initial device

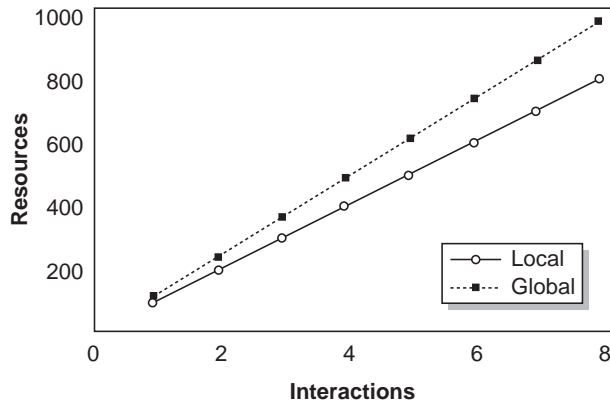


Figure 14. Resources versus interactions.

configuration, where many devices are scattered randomly within a circular area of a certain radius. Initially, all devices are set to transmit information at random intervals. After a short time, the devices shown in light color are transmitting, as shown at the top of Figure 16. Eventually, and based solely on local decisions taken by each device, the entire system begins to transmit in near unison. The graph at the bottom of Figure 16 shows the number of devices transmitting at once as a function of simulated time units. This graph indicates the time it takes to reach global synchronization, depending on the density and distribution of the devices and

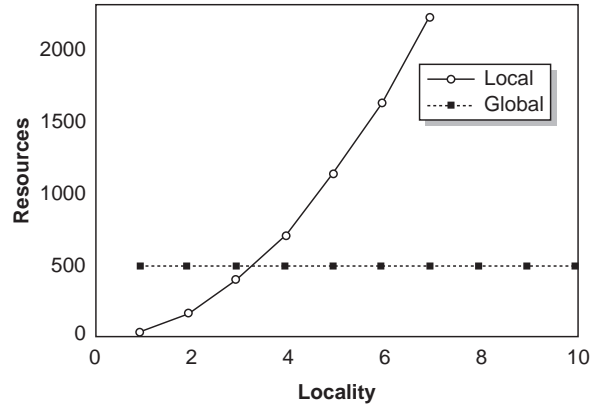


Figure 15. Resources versus locality in an emergent system.

their internal algorithm for determining when to transmit.

GE will attempt to minimize the amount of time required to reach synchronization. One goal is to reduce the amount of time to reach complete synchronization without adding overhead.

The example shown in Figure 16 has many interesting features that illustrate how GE will exploit emergence support of featherweight network services:

- All processing is local, and no global information processing or storage is required.

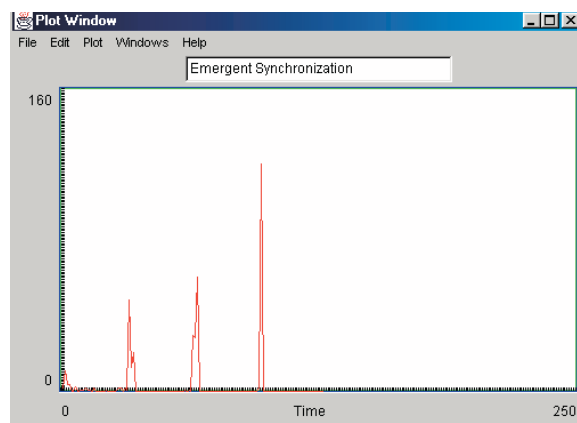
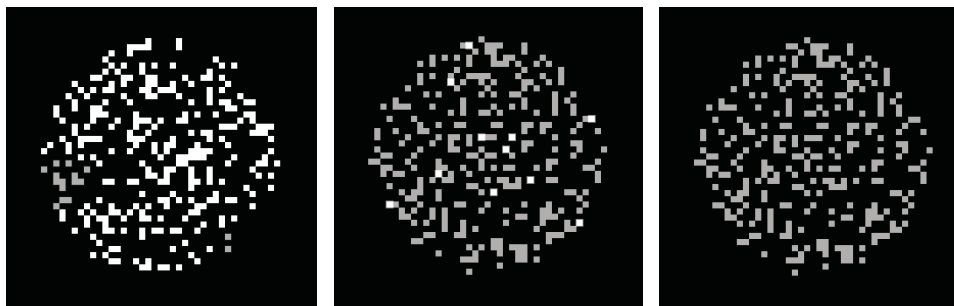


Figure 16. Emergent synchronization.

- System processing is minimal compared to the Internet protocol, because every device implements a simple algorithm that does not require exchange-of-state or communication with every device or central control node.
- The physics of the information should match the underlying communication network. For example, in a package sensor such as an accelerometer, many small abrupt movements result in information that has physical characteristics different from information representing long, smooth movements.

The success of this proposed work will be measured by the enhancement in network performance that emergent protocols exhibit over the best known nonemergent protocols.

EMERGENT MODELING

GE has developed a mechanism for injecting network component models directly into an operational network in order to improve the performance of the communications system^{2,4,6} using active networking. Given the weak processing power of remote sensors, the focus of this effort is on studying the feasibility of the inverse concept—a system of MEMS devices whose emergent aggregate behavior models a single MEMS device. An individual MEMS device will improve its performance using information from this emergent system model. This means the whole will reflect the part and, further, the part can somehow query the whole, as shown in Figure 17.

GE has developed a mechanism for injecting network component models directly into an operational network in order to improve the performance of the communications system.

While the processing power of an individual device is too weak to support a model on its own, an emergent model forms from the aggregate device behavior in the upper portion of the figure. This model is of either a single device or a group. In previous GE work,^{2-4,6} model behavior runs continuously ahead of time within the network. In this task, it is recognized that the weak processing power of MEMS devices will require emergent models to form for brief periods under fault conditions in order to help devices mitigate the impact of faults.

STEPS TOWARD HARNESSING EMERGENCE

The challenge of this proposed effort is to identify emergent protocols that arise from device operation and its interaction with other MEMS devices, especially in an environment of suboptimal connectivity, where messages can be lost and the system

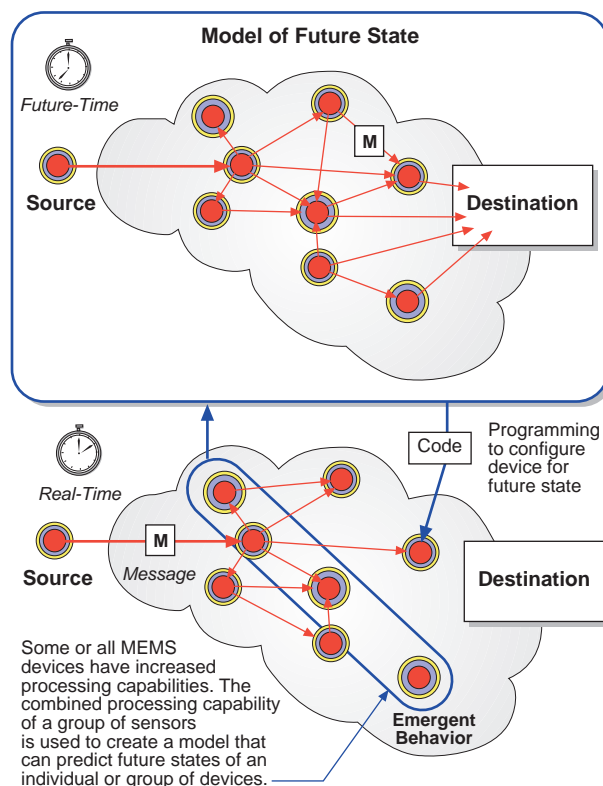


Figure 17. Emergent modeling.

operates on partial knowledge of its environment. By understanding the emergence of new behaviors in such an environment, novel network services will be developed.

Physics of information and understanding device behavior

The development of fundamental relationships between the physical environment and MEMS device computation and communication mechanisms involves harnessing inherent and ambient processing, accomplishing desired computational and communication specifications (Figure 18). Network services that foster efficient communication with minimal resource consumption are desired, as shown in Figure 19.

Emergent protocol analysis

Results will be extended toward understanding highly complex and emergent protocols for communication of MEMS devices, as shown in Figure 20. Understanding how MEMS devices behave when driven toward emergence will involve heavy use of simulation, such as the Swarm simulation.

Implementation of emergent protocols

Results will be extended toward harnessing inherent emergent protocols and implementing network

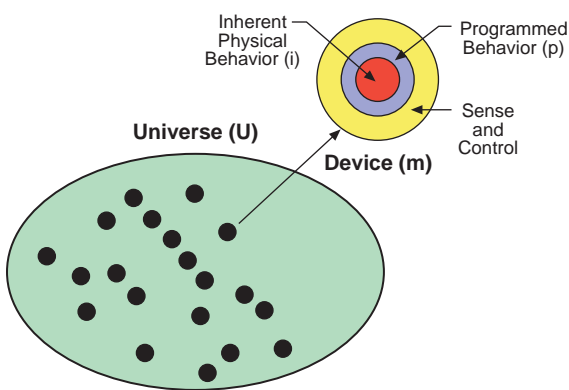


Figure 18. MEMS environment. Programmed and inherent behavior.

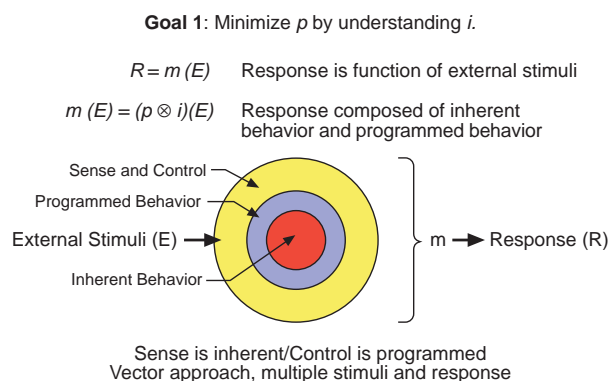


Figure 19. Device response.

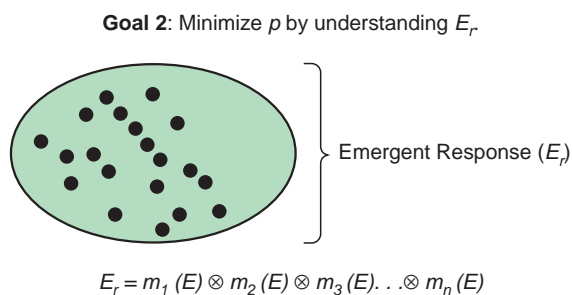


Figure 20. Identification of emergent response.

services to minimize the amount of computation and bandwidth required to achieve desired goals. Recognizing and controlling, or at least inducing, the desired synergistic behaviors explored to design and implement new network services with minimal MEMS device processing must be understood. Mechanical recognition of certain types of emergent protocols in systems will be required.

Simulation and experimentation

Emergent modeling and simulation tools will be used to mimic typical MEMS devices. Scenarios of the operation of interacting physical devices will be developed. Interaction behavior will be studied to define service protocols and control algorithms. The evolutionary body of progressively more complex simulation results will begin with a Swarm model of the physical entities. It will progress toward more useful emergent protocols that take advantage of fine-grain synergy between physics and information. The final simulation results should be an approximation of the communications portion of the prototype system. Simulations and emulations using complexity probes can be used to search for new forms of emergence and to experimentally validate controlled emergent protocols.

Emergent protocols for expanded network services

Results will be extended to design and implement emergent protocols for multicast and group communication services. The experience gained from implementing novel routing algorithms that exploit emergent protocols will be used to design and develop featherweight multicast and group communication services for networks of MEMS devices. These networks form a dynamic topology and have inherently unreliable communication among them. Table 1 shows some expected potential results.

Table 1 Expected communication benefits

Traditional network protocols	Potential new results
• Link (best effort vs. reliable)	• Inferred communication
• Routing	• Fault tolerance through fractal replication
• Forwarding	• Harnessing ambient processing
• QoS (real-time vs. non-real-time)	

Emergent modeling for enhanced network performance

GE has developed a mechanism for injecting network models into an operational network in order to improve the performance of the communications system.³ Given the weak processing power of MEMS devices, the feasibility of the inverse concept is the focus—a system of MEMS devices whose emergent aggregate behavior models a single MEMS device. An individual MEMS device will improve its performance using information from this emergent system model.

CONCLUSION

Blindly applying current communication and computation technology on MEMS devices would be fighting a losing battle against nature. The proposition this white paper hoped to reinforce in the

readers' mind was that MEMS devices can be more efficiently engineered by working with, instead of against, the environment in which they are placed. Specifically, two approaches were proposed for revolutionary gains in MEMS device communication. The first was to view all network devices as computational or active devices. Computation can take many forms. The amount of computation may vary, but every device has some type of computation, either programmed or ambient. Use of computation in an optimal manner is the same challenge faced by active networks. Thus, advances in active networks and networks of MEMS devices are mutually beneficial. The second approach was to optimize networks of MEMS devices via exploiting emergence. Understanding emergence requires understanding complexity; that relationship was touched upon relative to networking in this white paper. Use of emergence shows promise as a means to precisely engineer desired characteristics into systems of MEMS devices resulting in reduced size by removing unnecessary computation and control.

REFERENCES

1. Bush, Stephen F. and Kulkarni, Amit B., "Proactive Network Management Using Active Networks," CRD Technical Report 2000CRD112, http://www.crd.ge.com/crd_reports/index.htm
2. Bush, Stephen F., "Active Virtual Network Management Prediction Project Final Report" funded by DARPA/ITO Contract Number: F30602-98-C- 0230 supported by the Air Force Research Laboratory/IF.
3. Bush, Stephen F. and Kulkarni, Amit B. Active Networks and Active Virtual Network Management Prediction: A Proactive Management Framework. ISBN 0-306-46560-4. Kluwer Academic/Plenum Publishers. Spring 2001.
4. Bush, Stephen F., Active Virtual Network Management Prediction, Parallel and Discrete Event Simulation Conference (PADS) '99, May 1999.
5. Bush, Stephen F. and Barnett B., A Security Vulnerability Technique and Model, GE Corporate Research and Development, Technical Report 98CRD028, January 1998.
6. Bush, Stephen F., Kulkarni A., Galtier V., Carlinet Y., Mills K. L. and Ricciulli L., Predicting and Controlling Resource Usage in an Active Network, DARPA Active Networks PI Meeting December 6-9, 2000, Atlanta, GA.
7. Bush, Stephen F., Evans, Scott, Complexity-Based Information Assurance, To be submitted to Eighth ACM Conference on Computer and Communications Security (CCS-8), November 5-8, 2001, Philadelphia, Pennsylvania, USA.
8. Coore D." Botanical Computing: A Developmental Approach to Generating Interconnect Topologies on an Amorphous Computer," Ph.D. Thesis, MIT Dept. of Electrical and Computer Science, Dec. 1998.
9. J. P. Crutchfield. "The calculi of emergence: Computation, dynamics and induction," *Physica D.*, vol. 75, no. 1-3. Pages 11-54, 1994.
10. James P. Crutchfield and Melanie Mitchell. "The Evolution of Emergent Computation," Santa Fe Institute. 1994. Number 94-03-012.
11. Evans, Scott, Bush, Stephen F., and Hershey, John, Information Assurance through Kolmogorov Complexity Accepted for publication at the DARPA Information Survivability Conference and Exposition II (DISCEX-II 2001) to be held 12-14 June 2001 in Anaheim, California.
12. David A. Fisher and Howard F. Lipson. "Emergent Algorithms: A New Method for Enhancing Survivability in Unbounded Systems," Email: dfisher@cert.org and hfl@cert.org CERT@ Coordination Center Software Engineering Institute.
13. Edward Fredkin and Tommaso Toffoli. "Conservative Logic." *International Journal of Theoretical Physics*, vol. 21, pp. 219-53, 1982.
14. Harmon, S. Y., A Physical Model Of The Behavior Of Information Systems, DARPA Final Report, Volume I, Zetetix, Oak Park, CA, October 2000.
15. Kirchner W., Li M., and Vitanyi P., The Miraculous Universal Distribution, *The Mathematical Intelligencer*, Springer-Verlag, New York, Vol. 19, No. 4, 1997.
16. Amit Kulkarni & Gary Minden. "Active Network Services for Wired/Wireless Networks," INFOCOM'99, NY, March 1999
17. Kulkarni, A. B. and Bush, Stephen F., Active Network Management, Kolmogorov, Complexity, and Streptichrons. GE-CRD Class I Technical Report 2000CRD107 (www.crd.ge.com/crd_reports/index.htm).
18. Kulkarni A. B., Minden G. J., Hill R., Wijata Y., Sheth S., Pindi H., Wahhab F., Gopinath A., and Nagarajan A, Implementation of a Prototype Active Network, OPENARCH '98, 1998.
19. M. Mitchell, J. P. Crutchfield and P. T. Hraber. "Evolving Cellular Automata to Perform Computations," *Physica D.* 1993.
20. Ming Li and Paul Vitanyi, Introduction to Kolmogorov Complexity and its Applications, Springer-Verlag, 1993. ISBN 0-387-94053-8.

-
21. Rose, M. T., *The Simple Book, An Introduction to the Management of TCP/IP Based Internets*, Prentice Hall, 1991.
 22. Smart Dust Performer: University of California, Berkeley, Dr. Kristofer Pister (510) 643-9268
Agent: AIC Mr. Roy Peters (520) 538-409.
 23. Tennenhouse, D. L., Smith, J. M., Sincoskie W. D., Wetherall, D. J., and Minden, G. J., A Survey Of Active Network Research, *IEEE Communications Magazine*, 35(1): 80-86, Jan. 1997.
 24. Tinker P. and Agra J., *Adaptive Model Prediction Using Time Warp*, SCS '90, 1990.
 25. Wallace, C. S. and Dowe, D. L., Minimum Message Length and Kolmogorov Complexity, *The Computer Journal*, Vol. 42, No 4, 1999.
 26. Wojciech Zurek. *Complexity, Entropy and the Physics of Information*. Addison-Wesley, 1990. ISBN 0-201-51509-1.
 27. <http://www.swarm.org>