



Active Network Management, Kolmogorov Complexity, and Streptichrons

A.B. Kulkarni and S.F. Bush

2000CRD107, December 2000

Class 1

Technical Information Series

Copyright © 2000 General Electric Company. All rights reserved.

Corporate Research and Development

Technical Report Abstract Page

Title Active Network Management, Kolmogorov Complexity, and Streptichrons

Author(s) A.B. Kulkarni **Phone** (518)387-6827
S.F. Bush 8*833-6827

Component Electronic Systems Laboratory

Report Number 2000CRD107 **Date** December 2000

Number of Pages 17 **Class** 1

Key Words active networks, emergence, network management, complexity

This report discusses the goals and requirements that drive the architecture for an active network. The active network management framework refers to the minimum model that describes components and interactions necessary to support management via an active network. An overview of the management architecture for today's passive networks is discussed as a prelude to the presenting a new active network management model. Finally, the new active network management model is developed a step further into the initial exploration of emergent behavior within an active network.

Manuscript received November 20, 2000

Active Network Management, Kolmogorov Complexity, and Streptichrons

Amit B. Kulkarni and Stephen F. Bush*

October 6, 2000

1 Abstract

This document discusses the goals and requirements that an active network management framework should facilitate. The active network management framework refers to the minimum model that describes components and interactions necessary to support management within an active network. This is motivated by comparing management in current networks with the possibilities enabled to support management within active networks. Towards this objective, an overview of the current network management model is discussed as a prelude to describing and discussing a model for management in active networks.

2 Introduction

In the current communications model, managed devices are viewed abstractly as protocol layer two and protocol layer three network devices that forward data from source towards destination end-systems. The actions taken by these devices are predefined and fixed for each protocol layer and packet type as shown in Figure 1. The non-active data packets are shown transporting management requests to the managed device and a possible management response is shown leaving the managed device.

The current management model, as illustrated and implemented by such protocols as the Simple Network Management Protocol (SNMP) and the Common Management Information Protocol (CMIP), requires that network devices have a management agent that responds to management requests. Devices must be addressable and respond directly to remote management commands. The model assumes that network nodes are instrumented with the ability to respond to requests for pre-configured data points of management information. This requires

*This document is GE Proprietary.

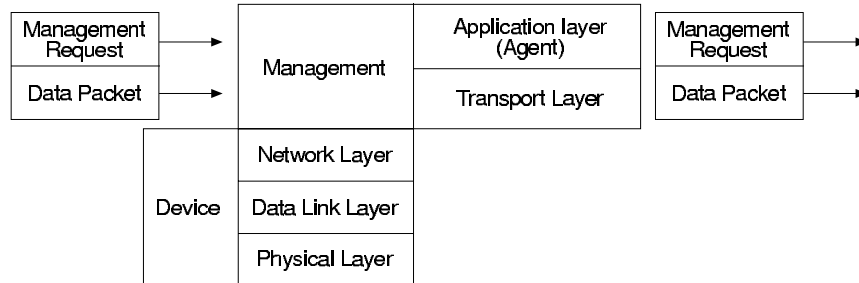


Figure 1: Current Management Model.

more than just the bottom two or three standard protocol layers. Therefore, management has never been a natural fit to the current non-active communications model for intermediate network devices. It was initially considered difficult and uncommon for any type of standards based management to exist because of the large number of non-interoperable proprietary attempts to solve the problem. Thus, the goal had been to implement a standard management framework that was robust and would be ubiquitously deployed across the Internet. SNMP had filled this role to some extent, however, active networks can allow for a better solution.

3 Motivation

The current network management model presents a rigid, statically-defined framework that is not so well-equipped to handle the needs of a bludgeoning Internet. The framework has several inherent characteristics that hinder extensibility and flexibility of the model. Yemini et. al. have listed some of the drawbacks of the current paradigm. Some important characteristics are described below:

3.1 Static Management

In the current management model shown in Figure 2, high-level queries are entered into, or generated from, a central management station that breaks the query down into low-level requests for data from managed entities. The current management model requires that all data values that would be needed for management must be predetermined and pre-defined in a repository of management variables called a Management Information Base (MIB). Each data point has a predetermined type, size, and access level and is called a Management Information Base or MIB Object. The result is that the MIB contents, that is, the collection of MIB Objects, must be painstakingly designed and agreed upon far ahead of time before they can be widely used. Even after accomplishing this,

elements of the MIB have static, inflexible types. This is antithetical to the objective of the active network framework, which seeks to minimize committee-based agreements. In an active network framework, elements of the Management Information Base have the potential to be dynamically defined and used by applications. The static data type of a MIB element may be reasonable for network hardware, but becomes less appropriate as higher layers of the protocol stack and applications are instrumented.

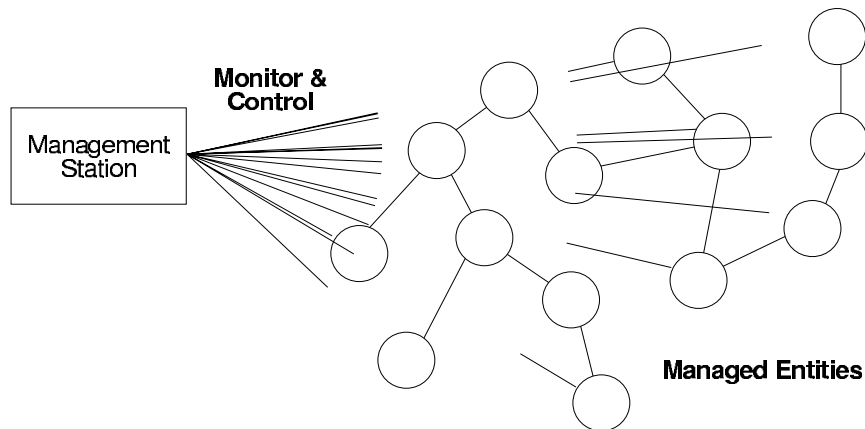


Figure 2: Current Centralized Management Model.

3.2 Poor network control architecture

The current management model leads to a poor network control architecture. Large delays are incurred as agents send raw data to a central management station that takes time to refine and process the primitive data and perhaps respond with a Simple Network Management Protocol *Set Request* control action. However, the current management model has been primarily concerned with monitoring rather than control, in part because control has been hampered by the long transfer delay times to the centralized management station. While management *Set Requests* can be used for control purposes, few MIBs today utilize the set commands for any type of real-time control.

As SNMP in current non-active networks has made steps toward providing reliable, integrated network management, the demand for more systems integrated management and control increases. Perhaps the demand is fed by the success of SNMP and by the explosion in the size of communication networks and number of applications utilizing them. Network administrators are pushing to extend network management ever higher towards and into the application layer. Integration is a primary driver. Clearly, applications, end-systems and

the network all need to be managed in an integrated fashion.

3.3 Limited Instrumentation

The paradigm of instrumenting network elements is not the best solution for managing higher layer protocols and applications, especially in an active network wherein applications have a direct interaction with network elements. One reason is increased complexity. Network hardware devices and low-level network protocol layers behave in precise, well defined ways. On the other hand, active applications can interact with network protocols and other applications in myriad ways. This complexity in interaction requires a proportional increase in the number of management data points. Instrumenting every network device and end-system to support management of every application is not a feasible or scalable option. Building network agent capability into every application is also not a scalable or feasible option. This could lead to other problems, such as redundant management data points, because two applications interacting with each other utilize the same management agent capability.

3.4 Centralized management

Another characteristic of the current management paradigm is that intermediate nodes are not designed to support management algorithms on the nodes themselves. However, fully integrated system management has always been the goal. Thus, values from data points from all managed elements, including applications, are simply transported to a centralized management station where all refinement and processing takes place. Proxy agents are sometimes used to manage devices that have non-standard or non-existing management interfaces. Proxy agents serve as intermediary translators between the management standard and the operations of which the device is capable for management. The only other place that processing could be done within the network is in the managed object's agent. However, the prevailing philosophy has been that the managed object should be fully devoted to its primary task of forwarding data, not management, and thus the agent is designed and implemented to be as simple and efficient a process as possible. The agent simply responds to requests for management data point values and generates unsolicited trap messages, hopefully, infrequently and only under extreme conditions.

3.5 Goals of active network management model

The new active network management goals should consist of:

- Automated self-management and control of applications.
- Ability to dynamically add/remove management features across all active applications.

- A richer integrated management view of the network and applications than in the old network management model.
- Decentralized and distributed management within the network for increased efficiency.
- Extremely reliable in the face of network failure.
- Support for integrated management of legacy applications

A few words of explanation are in order to justify why these goals are worthy of pursuit. Clearly, network management benefits from being as automated as possible. The words “self-management” are used because it is assumed that the system is able to determine best how to manage and control itself. An integrated view is the most concise and logical for human consumption and allows quick identification of correlated events. This assumes that a security policy mechanism is in place for network managers to gain access only to their own views of the system. The goal is that active networks will allow a richer semantic view of the integrated system. We want the system to be decentralized and distributed since that provides the most efficient use of resources and better response times. In addition, it can allow for graceful degradation of performance as resources fail. Finally, management is most critical when the system is failing. Thus the management system must be as robust and reliable as possible; that is, it should be the last service to fail. A framework within active networks that supports these goals is useful. However, care must be taken in developing a framework that does not preclude the development of general-purpose innovative management techniques enabled by active networking.

4 Impacts of an active network management model

Active networking affords an opportunity to take a new look at the network management problem and communications in general from a different perspective. It is a perspective that flips the traditional networking paradigm on its head. By allowing general-purpose computation on traditional intermediate network systems, it is no longer required that application processing, including network management processing, be restricted to end-systems. Optimum management efficiency can be achieved because processing can be allocated to intermediate network resources. This allows for a larger set of feasible solutions to the allocation of processing resources. For example, the old philosophy of keeping communications as simple possible has resulted in a plethora of highly specialized protocols illustrated by the large number of Internet Engineering Task Force Request for Comments that are extant. In terms of network management, the old philosophy has caused enormous inefficiency by requiring large amounts of data to be transported to centralized management stations, even in instances when the data turns out to be of limited or no value. The active network

model provides a communications model that is a better fit to the management model. In the active network reference model, intermediate-system active devices have the ability to accept and process any packet as a natural part of its packet processing, including network management packets. In fact, in the new management paradigm, management can be integrated into the processing framework itself, that is, packets are the application and manage themselves.

The flexibility and extensibility of the active network management model has numerous impacts on the way management is performed in the future. Active networking enables novel features and strategies that are not possible in the current passive model. This section describes a few important impacts.

4.1 Dynamic application-oriented management

In the new management model shown in Figure 3, it is possible for a high-level management query in executable form to be sent directly to the managed active application. Because the managed application is active, it is implemented via active packets. The management query active packet interacts with the active application's packets in order to determine the result of the query. Given active network protocol composition, methods are dynamically bound to an application that no longer require a MIB with static data point definitions to return predefined values, but instead, access local data points, compute a result from the local data and return only the final result, or some set of data culled from the local data that can lead to the final result, which may be computed in another part of the network. Many management systems today operate by polling a value and setting a threshold that trips an alarm when the threshold is crossed. Frequent queries result in wasted bandwidth if the threshold is rarely reached. The only information required in such cases is the alarm. In the active network management environment, the threshold crossing detection can be dynamically bound as a method in the managed active application.

4.2 Improved Control Architecture

The current management philosophy requires that a SNMP *Set Request* be used for control purposes. This results in long delays when the controller is a centralized management station as compared to the active network model that enables local computation and control. Delays are clearly dangerous in a control system. Thus, using the SNMP *Set Request* is also highly inefficient for dynamic control purposes. Active networks allow more distributed control for management purposes than the current management model and provides an opportunity for a new management paradigm. The control algorithm is bound directly in the managed active application. Problem diagnosis and isolation can possibly occur locally thus reducing the delay incurred by dealing with a centralized management station.

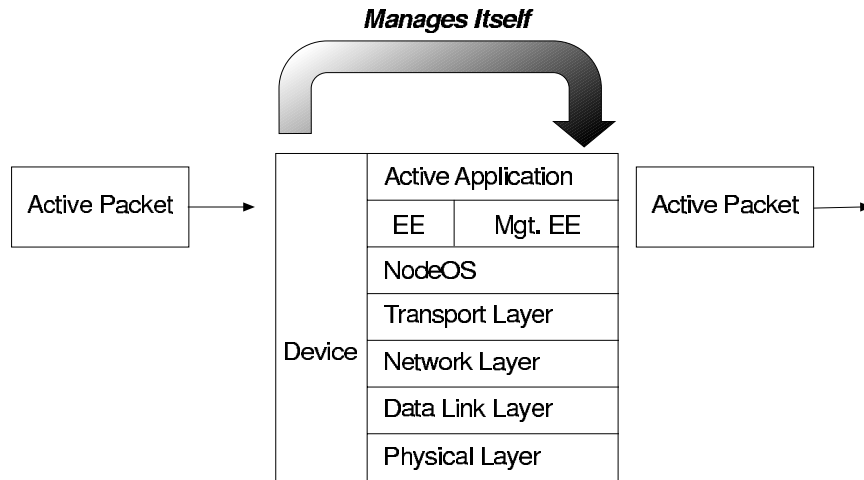


Figure 3: Active Management Model.

4.3 Pro-active Management

The active nature of the network also allows a framework in which efficient prediction of network behavior is possible. The Active Virtual Network Management Algorithm [2] takes advantage of the active network to provide a model-based predictive management control framework. This requires a form of introspection that is possible in the new management model. Introspection is enabled because applications can control and manage themselves to a greater degree with active networks than ever before in the old management philosophy. The following example shows data in an active network management model has the ability to be queried by standards based network management protocols. A small SNMP agent is encapsulated with the active data as shown in Figure 5. When the data is queried the agent responds with the MIB values maintained by that specific agent's MIB. The converse of this is shown in Figure 4, that illustrates active data containing a management client capable of querying management agents. This concept has been proto-typed in Java in the active network testbed at General Electric Corporate Research and Development.

It has been recognized that, even in the new active management model, systems administrators require an integrated view of the entire managed system. However, note that integrated does not necessarily imply centralized. Also, note that the functionality of an integrated management view has changed dramatically from the old management model. The current management view consisted of displaying values from static MIB data types that are predefined. This is in contrast to the new management model that consists of controlling the algorithms (methods) that are bound to managed active entities and displaying

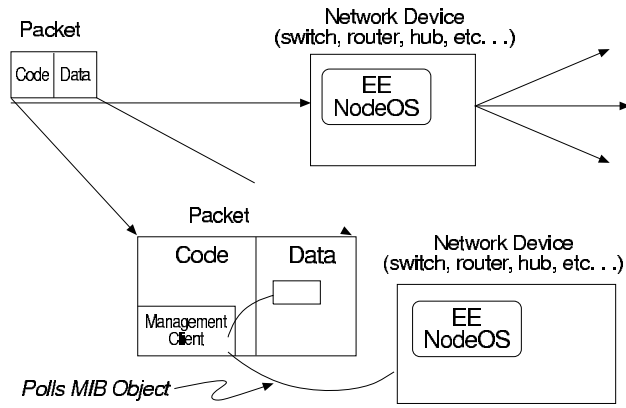


Figure 4: An Overview of the Traveling SNMP Client.

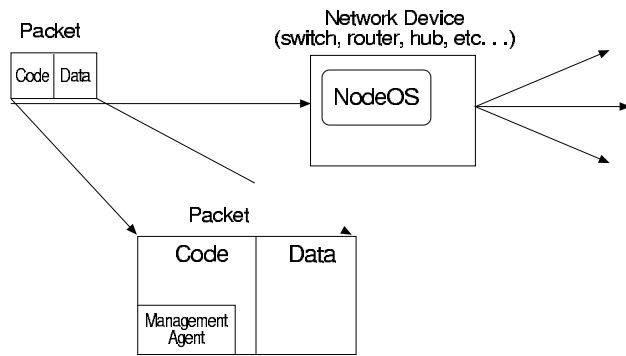


Figure 5: Queryable Data.

results from those algorithms. Thus, the new management model deals with methods rather than data types. The Active Virtual Network Management Prediction Algorithm is a step towards an active network management framework by enabling model-based predictive control as discussed in [2].

5 Towards an Active Network Management Framework

The previous section discussed the goals of a new framework for network management. Consider what is required from the framework in order to achieve these goals. Automated self-management and control of applications require application developers to provide monitoring and access into their applications. While an application may be self-managing and autonomous, it cannot be a completely closed system. The application needs information about other applications and the network that it resides upon. The application may need to negotiate with other applications for resources. The management interface between applications could be accomplished through MIB definitions as is the case in today's non-active networks, however, more is possible with an active network. For example, the MIB could itself become an active entity. Model-based predictive control is a particular mechanism enabled by the Active Virtual Network Management Prediction Algorithm described in detail in [2]. A fully autonomous, self-managed application requires:

- Inter-application semantic specification
- Inner-loop control mechanisms
- Negotiation capability
- Managed data semantic correlation
- Security policy

The negotiation capability and inter-application semantic specification are of primary interest here because they require some form of semantic knowledge and goal seeking capability. While dealing with semantic knowledge and goal achieving research are major efforts in their own right, the new management model architecture should facilitate and encourage their development. The integrated management view requires that all the management information from each managed entity be brought together and presented to a single user. This means that a policy must be in place to control access to information and the data must have the ability to correlate itself with other data for an integrated view. This requires a security policy and managed data semantic correlation.

There are several spheres of management in the active network management model; namely, the Execution Environment (EE), the Active Application

(AA) and the Network management algorithm, where a network Management Application (MA) is a new management feature to be added to all Active Applications (AA). The ability to dynamically bind methods into active applications is an assumed feature in active networks. The actual mechanisms for inserting methods into an existing and executing application are discussed in [7]. A brief summary of example methods are illustrated in Table 1. Self-organizing management code, knowing when, where, and how to insert itself into the managed active application is a goal that is partially met by the Active Virtual Network Management Prediction Algorithm. The Active Virtual Network Management Prediction framework discussed in detail in [2] demonstrates the fundamental requirements of the new active network management framework, namely:

- Access to managed device monitoring and control.
- Insertion of monitoring elements into arbitrary locations of active applications.
- Injection of executable models onto managed nodes and/or into managed active applications.
- Injection/interception of management packets within the network.

Composition Type	Reference
Functional	[4]
Dataflow	[3]
Slots	[6]
Signaling Extensions	[1]

Table 1: Active Network Composition Methods.

6 A Network Wide View of Distributed Active Network Management

A truly distributed, automated, and self-managed network requires network applications that include management functionality in their design and deployment. In order to avoid duplication of network application management implementation throughout the network, good design dictates that the management functionality is modular, indeed, separate from the application both in time and space, thus minimizing the overhead of redundant management applications that are not needed in every instant of time or in every location. This necessitates the design of management applications that have the ability to form when and where they are required. Formation should be done in a completely distributed and globally optimal manner.

An approach towards testing the design of such a solution in Swarm is described in this document. Swarm is a software package for multi-agent simulation of complex systems originally developed at the Sante Fe Institute. Swarm allows the simulation of large numbers of concurrently interacting agents. In Figure 6 heat is generated at potential trouble spots within the network that attract the proper management entities which compose in such a manner as to best solve the problem. The mechanism of problem identification, attraction, and composition is transparent to the application developer and requires minimal overhead. Thus, simple cooperating entities are developed as shown in the Swarm display depicted in Figure 6. The square entities in Figure 6 are management objects and the ripple patterns represent heat. The Swarm simulation package provides the ideal environment within which to examine the ability of various solutions to meet the requirements of low overhead, fast problem identification and composition, and accuracy.

7 Streptichrons and Complexity

In [2] the concept of streptichrons was introduced as active packets expected to exist in the future that carry executable code necessary to represent future behavior. The executable code necessary to represent future behavior was casually assumed to be designed in a more compact form than transmitting the equivalent static, non-executable data in a piecemeal fashion. Clearly, the algorithmic nature of the streptichron allows for more compression. As a simple example, a million digits of π can be transmitted, or more compactly and simply the description of a circle and the command to divide the circumference by the diameter. This document examines the manner in which active network management can benefit from Algorithmic Information Theory enabled by Active Networking. This is a relatively over-looked area in active networks. Because of the new paradigm and enhanced capabilities of active networks, this work proceeds along the lines that a radical new perspective in understanding network management should be taken. Complexity and Algorithmic Information Theory [5] are utilized to gain a new understanding of network health, behavior, and maintenance.

7.1 Active Network Management in Action

The objective of this work is a step towards communication networks whose inherent state, or natural tendency, is optimum performance. Faults should naturally attract the entities required to eliminate faults. The proposed mechanism to accomplish this can be summarized in three steps.

The first step is problem detection; the fault must identify itself to the entities capable of eliminating the fault. Notice that this does not necessarily require that a human identify or understand the fault, only that the solution

entities be capable of recognizing it.

Once the fault is identified, the next step requires that the information be efficiently and accurately propagated to the solution entities with minimal overhead to the network. Once a fault occurs, the network may already be impaired; adding additional overhead in trying to solve the problem would only exacerbate the problem. To re-state this step from another viewpoint, the necessary and sufficient solution entities should be attracted to the proper locations to solve the problem.

The final step in this vision is that only the necessary and sufficient entities required to correct the fault arrive, and that these solution entities act in a cooperative manner to quickly and accurately correct the fault.

7.2 Complexity and Network Health

The work described in this document seeks new and better ways to represent network health and thus attempts to explore concepts other than network topology-based representations of network health. The specific mechanism proposed in this document takes advantage of Complexity Theory [5]. The complexity of a piece of information is the size of the smallest program capable of producing that piece of information. Thus, there are strong ties between Algorithmic Information Theory and Complexity. A truly random piece of information cannot be compressed and its length is its complexity. Clearly, Active Networks will facilitate the application of results from Algorithmic Information Theory. Complexity is, in general, uncomputable. However, bounds on complexity can be derived.

In order to introduce the concept, imagine a space filled with entities representing the values of various monitored objects from the managed system. As a specific example, each entity could be an SNMP object and its corresponding value. The location in space of each entity is **random** and each of these entities move in a random fashion. A truly random sequence is incompressible, thus, the sequence representing the location of these entities cannot be compressed.

Each entity has a normal operating range within which its value should fall during “normal” operation. As the operating range is exceeded, heat is generated. The other entities are attracted towards the heat, forming circular patterns, or clusters. Figures 6 and 7 show the entities and the heat generated from the entities after one hundred simulated time units. The pattern formation results in a loss of randomness and in a more compressible representation. As the magnitude and number of heat clusters increases, the more randomness is reduced and compressibility increased. The result is that more severe faults can be represented in smaller and more efficient forms for transmission. Thus a relationship between network health and complexity is established in this work. Figure 8 shows the cluster rate for a simulation with a specified probability of fault occurrence, duration, and severity. The measure of clustering is an attempt to measure the complexity. This work is pursuing better complexity

metrics. The cluster measure is one minus the proportion of the entity that is surrounded by adjacent entities. Note in Figure 8 that clustering clearly occurs. Each entity in this experiment had the same probability of fault occurrence and severity.

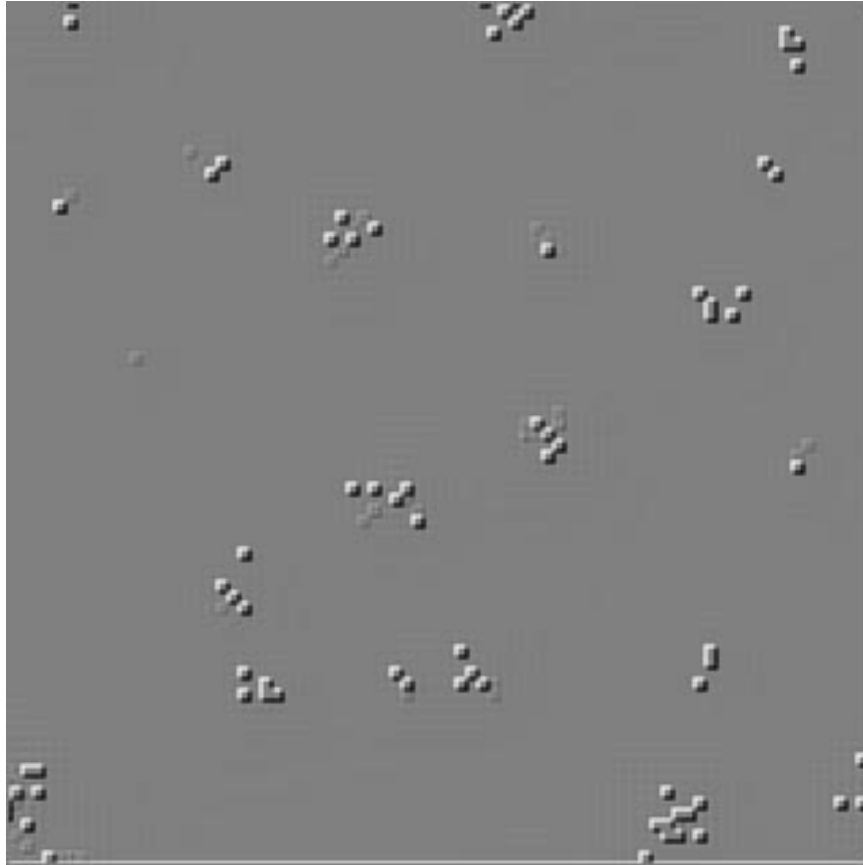


Figure 6: Entities with Low Fault Rate and Severity.

7.3 Propagation of Network Health

As established in the previous section, network health is more compressible as it deteriorates, allowing severe conditions to be propagated more quickly and efficiently. However, a truly pro-active system solves problems before they occur. Active Virtual Network Management Prediction as described in [2] can be used to run the system forward in time within a virtual overlay network in order to predict network health. In addition, patterns that occur early leading to

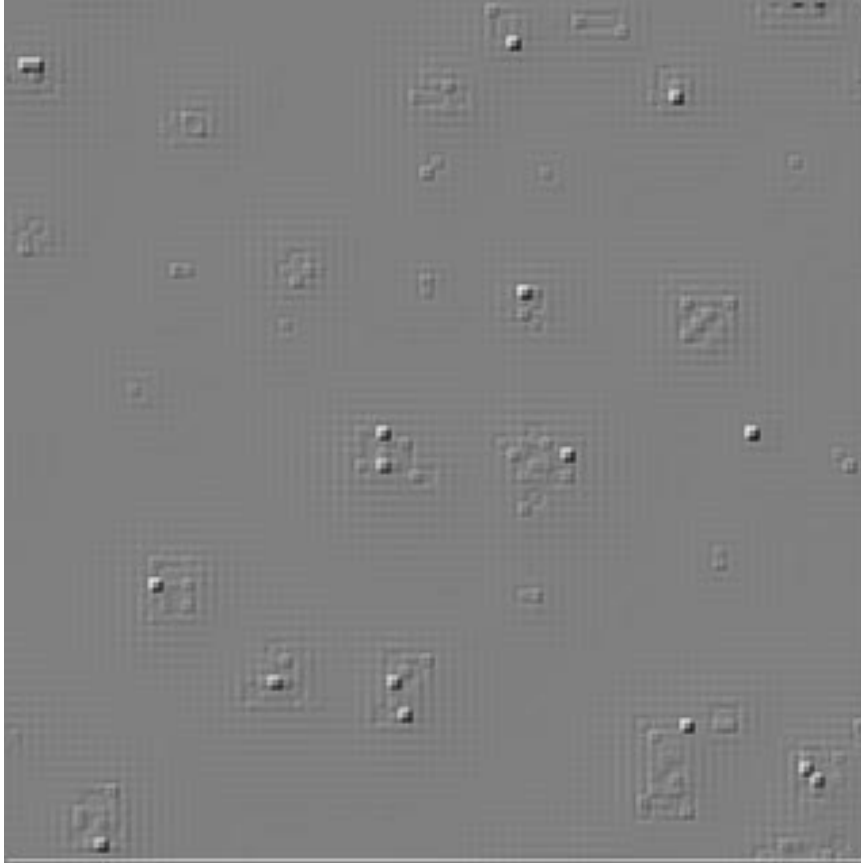


Figure 7: Entities with High Fault Rate and Severity.

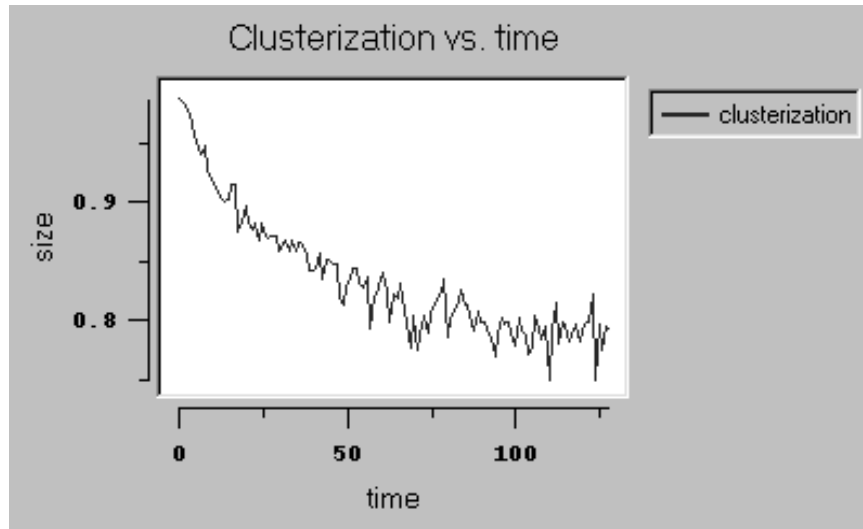


Figure 8: Cluster Rate for Low Fault Rate and Severity.

fault formation can be detected earlier. In either case, this state information is propagated through the system by a simple but efficient mechanism. This mechanism is simply the fact the each node includes the state of each of its adjacent neighbors' states within its health status. Thus, by a low overhead, adjacent exchange of state with each of its neighbors, each node contains a full representation, in compact form, of the entire network health.

7.4 Network Repair Entities

A key component of fault repair is the network solution, or repair, entity. This entity must support at least one type of repair function. It must also be capable of recognizing the faults that it is best at repairing. Finally, it must be able to cooperate with other repair entities to eliminate complex faults. The repair entity must be able to do this with a minimum of overhead and ease of development on the part of the application programmer.

It is assumed that the application developer knows the design and likely faults that will occur within the application. The developer designs repair entities, ideally one per fault and constructs the entities to be as simple as possible. The repair entities must include a sensor to provide feedback on the progress of the repair.

A transparent wrapper for the repair entity then “learns” to identify the faults it is most capable of handling. Note that the design proposed in this document does not assume any a priori indication by the developer to the repair entity about how to identify faults and in what manner it should cooperate with

other repair entities. The goal is for the wrapper to learn the optimal manner of operation.

The training sequence for a repair wrapper is always ongoing. Initially, the repair packet is attracted to all faults. As the repair entity receives feedback on the effectiveness of its efforts for various faults and in various cooperating combinations with other repair entities, it learns how to respond to future faults. It may learn that it is ineffective for some fault patterns, highly effective operating alone on other fault patterns, or effective cooperating with other repair entities on other fault patterns. Thus, the system adapts as new applications and new repair entities are introduced into the system. The adaptation occurs in ways in which human pre-programmed operation could not predict or comprehend.

8 Next Steps in the Experimental Validation

The second phase of the experimental validation is more interesting. The purpose of this phase is to determine the effectiveness of the repair entity learning capability. The proof-of-concept for repair entity learning proceeds by building Swarm representations of repair entities and faults. Notice that this phase of validation requires the results of phase one in which the fault identification clusters are formed. In this phase, the faults are represented as two dimensional matrices containing ones and zeros. The ones represent Swarm object locations in the fault identification phase from previous phase one executions.

This phase is concerned with the active wrapper learning capability, and not the details of the fault repair action. Thus, each instance of a Swarm repair object is pre-programmed to be effective in randomizing certain portions of the fault identification matrix. The repair entity monitors its affect upon the fault identification matrix. The information monitored includes the change in the fault identification matrix and the types of other repair entities currently acting upon the fault. Thus the repair entity can determine its incremental impact upon the fault repair as well as the effectiveness of the repair group as a whole. Points in the cluster which are reinforced over time remain as learned information, while outliers are eventually dropped from memory.

The expected results from this phase of the experiment are that only the necessary and sufficient repair entities required to correct a fault are attracted to that fault. Also, the system will adapt as new repair entities and new types of faults are introduced into the system. The key performance measures in this phase are: As the same type of fault repeatedly occurs in the system, fewer and fewer entities will respond until only the necessary and sufficient entities respond. Key measurements will include the reduction in the number of repair entities which respond over time as well and the time to repair a fault. It is expected that repair time and overhead in terms of repair entity movement will decrease over time. The final validation step in this phase is to introduce new types of faults into the system and repeat the repair response time and overhead

measurements.

References

- [1] Bob Braden, Alberto Cerpa, Ted Fischer, Bob Lindell, Jeff Kaum, and Graham Phillips. Introduction to the ASP Execution Environment. Technical report, USC/ISI, February 2000. url: <http://www.isi.edu/active-signal/ARP/index.html>.
- [2] Stephen F. Bush. Active Virtual Network Management Prediction. In *Parallel and Discrete Event Simulation Conference (PADS) '99*, May 1999.
- [3] Sushil da Silva, Danilo Florissi, and Yechiam Yemini. Composing Active Services in NetScript. In *Proceedings of the DARPA Active Networks Workshop (Tucson, Arizona)*, March 1998.
- [4] Michael Hicks, Pankaj Kakkar, Jonathan T. Moore, Carl A. Gunter, and Scott Nettles. PLAN: A programming language for active networks. *ACM SIGPLAN Notices*, 34(1):86–93, January 1999.
- [5] Ming Li and Paul Vitanyi. *Introduction to Kolmogorov Complexity and its Applications*. Springer-Verlag, August 1993.
- [6] Samrat Bhattacharjee, Kenneth L. Calvert and Ellen W. Zegura. Reasoning About Active Network Protocols. In *Proceedings of ICNP '98*, October 1998. Austin, TX.
- [7] Ellen Zegura. Composable Services for Active Networks. AN Composable Services Working Group, September 1998.

A.B. Kulkarni
S.F. Bush

**Active Network Management, Kolmogorov Complexity, and
Streptichrons**

2000CRD107
December 2000