18th ICCRTS

Using Autonomics to Exercise Command and Control of Networks in Degraded Environments

Topics
Topic 10: Cyberspace Management
Topic 7: Architectures, Technologies, and Tools
Topic 8: Networks and Networking

Authors

Phillip Verbancsics
Douglas S. Lange
Space and Naval Warfare Systems Center – Pacific
San Diego, CA 92152

Point of Contact

Douglas S. Lange
Space and Naval Warfare Systems Center – Pacific
(619)553-6534
doug.lange@navy.mil

Using Autonomics to Exercise Command and Control of Networks in Degraded Environments

Phillip Verbancsics
Douglas S. Lange

Space and Naval Warfare Systems Center – Pacific

Abstract

Autonomic approaches enable large, complex systems to exhibit self-adaptation in response to attack or rapid degradation of the environment. This paper applies one such autonomics approach, the Rainbow autonomics framework, to naval command and control systems. Rainbow employs an abstraction language that models a managed system, probes that read data from the managed system, gauges that interpret data from probes, strategies that adapt the managed system to changes, and actuators that effect the desired strategic changes on the managed system. Because Rainbow represents managed systems as architectural abstractions, varied systems can be modeled, including such naval systems as the Command and Control Rapid Prototyping Continuum (C2RPC), simulated groups of operational forces that include autonomous vehicles [1], and navy data centers. All three can be described in the abstraction models of Rainbow and all can be managed by an autonomics framework. The focus of this paper is on the effects of Disconnected, Intermittent, and Limited (DIL) connectivity environments on the capability of autonomics to manage a system in such environments. The results show that DIL environments have a negative effect on centralized autonomics' capability, such as Rainbow, in managing target systems.

1. Introduction

A critical research focus in the design of cyber-physical systems is *autonomic computing*, that is, software that allows such systems to self-heal, self-adapt, self-optimize, or self-defend [2, 3, 4]. This autonomics research is necessitated by the growing reliance on cyber-physical systems whose scale and complexity make it both difficult and costly for humans to manage. Autonomic algorithms typically include monitoring mechanisms that allow the managing software to observe the state of the managed system and the environment that contains the system, detection and analysis to understand the running behavior, and then effectors that effect required actions on the managed system when problems are detected. Disconnected, Intermittent, and Limited (DIL) connectivity network environments may present both an ideal and challenging domain to apply autonomics. In such domains, the changing environment (e.g. connecting/disconnecting components) requires that systems adapt and re-optimize in the face of environment changes. Furthermore, intermittency could mean windows of opportunity so short that a human is incapable of responding. In contrast, autonomic approaches can effectively exploit these short communication windows. Similarly, limited bandwidth could constrain how often data and commands can be sent, also resulting in reduced opportunities to effect changes in the managed system.

Autonomic approaches manage complex systems such that they exhibit self-adaptation in response to demands on the system or degradation of performance. This paper applies one such autonomics approach, the Rainbow autonomics framework [5], to naval command and control systems. Rainbow employs an *architecture-based self-adaptation* approach that models the managed system through an *architecture description language* (ADL), receives information system through gauges from probes that read data from the managed system, and strategies defined in the *Stitch* language that provide instruction on how the managed system should adapt to changes. Naval systems currently being researched include the Command and Control Rapid Prototyping Continuum (C2RPC), simulated groups of operational forces that include autonomous vehicles [1], and navy data centers. All three can be described in the abstraction models of Rainbow and managed by such autonomics frameworks. The focus of this paper is on virtualized server environments, such as those found in modern data centers or embodying future C2 systems like C2RPC. The effect of intermittent communications is demonstrated in the autonomic managements of virtualized resources in a server cluster. The overall effect of degraded communications is shown to be degraded performance in managing the system, with the more intermittent the communication, the greater the effect on management performance.

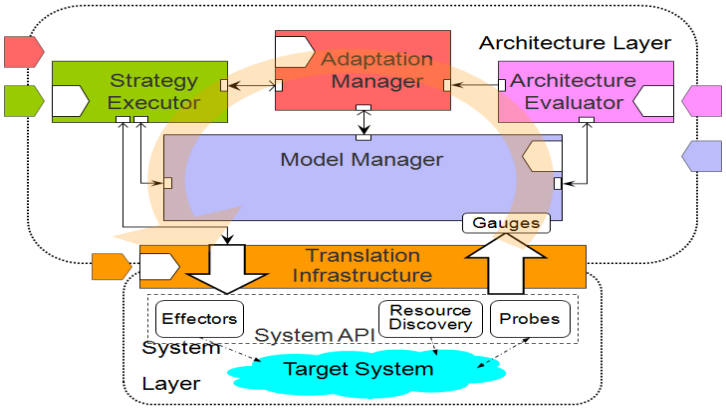

Image 1: The Rainbow Framework (from **[18]**) consists of four primary components: The Model Manager, the Architecture Evaluator, the Adaptation Manager, and the Strategy Executor. In addition, for each target system being managed, there is a Translation Infrastructure that provides the means to send data to the Rainbow framework and for Rainbow to effect changes in the target system.

2. Background

Beginning with the initial research from IBM [4], the core capabilities of autonomic systems have included: self-configuration, self-healing, self-optimization, and self-protection. Self-configuration is the capability for a system to setup and manage settings for running in the deployed environment without manual intervention. Self-healing is the ability of the system to recover from errors, faults, damage, etc, to software or hardware. Self-optimization focuses on allowing a system to change its configuration to improve performance. Self-protection allows a system to respond to attacks, such as denial-of-service, that could damage or disrupt the system if allowed to occur. While having all of these components are necessary in autonomic systems, each can have a role to play. Active research in the field of autonomics has produced varied approaches that address these self-* capabilities. One popular approach is rules-based systems that allow managed systems to adapt based upon a priori defined rules [6]. Other self-healing research has investigated software that modifies its own architecture during execution, known as runtime dynamism [7]. Another approach, inspired from research in artificial intelligence, is modeling systems in state-action pairs. In this way, the autonomic framework explicitly understands the transition between specified system states and the actions that cause such transition. Such approaches often model system in formal logic or predicate calculus [8, 9, 10]. Finally, significant research has investigated autonomics through formal modeling of system architecture [11, 12, 13]. A survey of the approaches is given in [14].

3. Approach

A major problem with many autonomics approaches is they are tied to particular system implementations making scalability an issue for complex systems. For example, being tied to a particular architecture means a lack of flexibility to express new additions that may have not been designed for. Rainbow [5], the method in this paper, addresses this problem through abstraction of architecture into a formal architecture description language, ACME. The ACME language allows Rainbow to express any system that can be expressed formally. The Rainbow framework maintains the ACME model description in its Model Manager (see Image). The Architecture Evaluator then analyzes the model to determine violations of constraints that were defined for the target system. If constraints are violated, then the Adaptation Manager is triggered, which evaluates various strategies defined in the Stitch language to determine the best approach to solving the problem that triggered the need for adaptation. The Adaptation Manager sends the chosen strategy to the Strategy Executor that is tasked with programmatically carrying out the strategy and effecting changes to the target system (see Figure 1). The Rainbow framework interacts with the target system through Probes, which send data to Rainbow, Gauges, which receive data from Probes, and Effectors, which are actionable items that cause changes on the target system. This paper builds upon the successful Rainbow framework, as described in the next section.

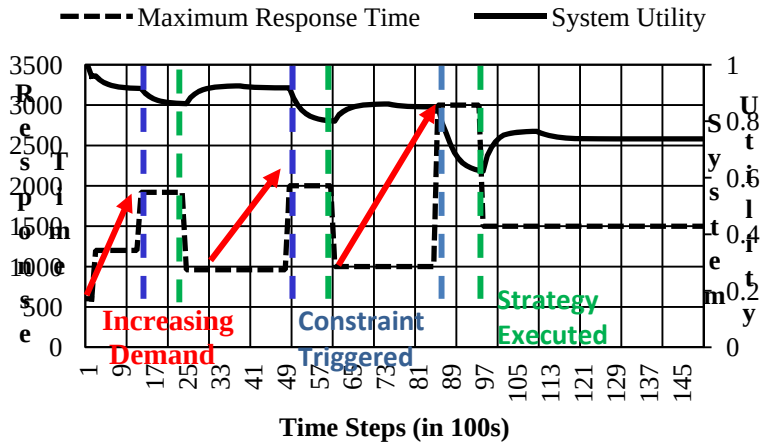

Figure 1: This example demonstrates the effect of Rainbow's autonomic management on the system. Based upon demand on the system, response time for services begin to increase. Eventually, the measured response time violates the constraints defined in the ACME model managed by rainbow. The Adaptation Manager then activates a particular strategy that addresses the increased demand and the Strategy Executor effects the desired changed in the target system, leading to a decrease in response time.

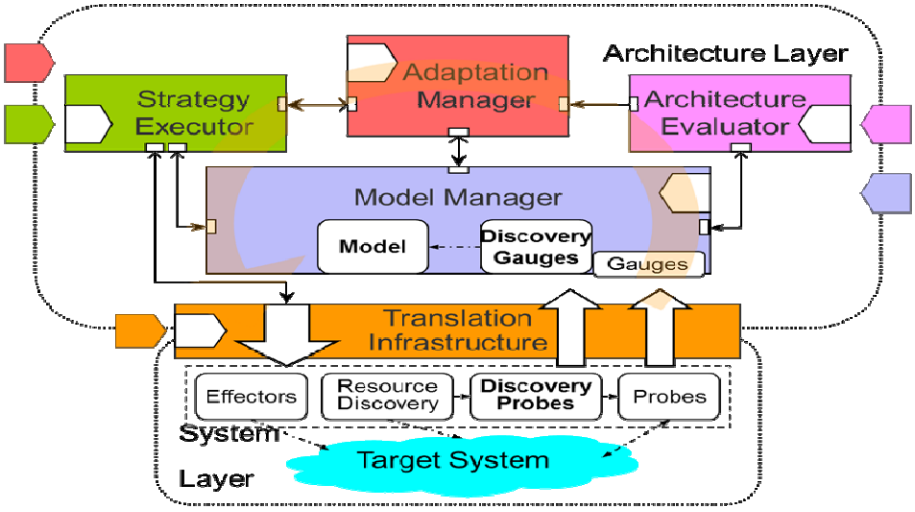

Image 2: Two components were added to the Rainbow framework to facilitate discovery. The first item is Discovery Gauges that directly interact with the ACME model to generate necessary model components to represent the discovered elements of the target system. The information for the creation of architectural elements comes from Discovery Probes, which are responsible for initializing the Probes that report data about the target system. In this way, whenever any component begins to report data, it first registers through discovery to generate matching architecture in the ACME model.

3.1 Rainbow Discovery Extension

This paper introduces an addition to the Rainbow framework. This extension is a key enabling approach for large-scale and dynamically changing systems, such as those that operate in DIL environments. This technology is the ability to add new components dynamically through a process of discovery for Rainbow to monitor. Many systems are designed with a particular, a priori specified architecture, or, must be manually updated [15]. However, discovery is an important abstraction technique that allows systems to dynamically reconfigure by providing metadata about individual components, such as in service oriented architectures [16].

To add this capability to Rainbow, a special Gauge-Probe pair is created (see Image 2). When a new component is going to start reporting data about itself to Rainbow, the Discovery Probes send metadata about the new component to the Discovery Gauges. This metadata includes the component's architecture description, gauges that will be needed to receive data, and other pertinent information to create the necessary Rainbow infrastructure for the system component. The Rainbow Discovery Gauges then process the received metadata and issue commands to the appropriate Rainbow framework sub-components for the creation of the necessary framework infrastructure to manage the discovered component.

4. Experimental Setup

The experiments in this paper are designed to investigate the effect of degraded environments on the performance of autonomic systems in the management of their target systems. In particular, the effect of intermittent communication of the performance of the Rainbow framework is examined, that is, if the autonomic system can only probe data and effect the system infrequently, then how does performance of the managed system change. To this end, the experiments described in this paper look at the performance of Rainbow on a simulated managed system where the interval between updates from the simulated system to the Rainbow autonomics framework is gradually increased thus simulating an increasingly degraded communication situation.

4.1 Degraded Environments

In these experiments, the simulated system is a set of servers with virtualized services in a service oriented architecture, representative of future C2 systems, such as C2RPC. These virtual machines service a workload that varies over time. In this simulation, the time varying workload reaches its peak at 1200 hours each day and its minimum at 0000 hours, a gross simplification of a normal workday. The Rainbow autonomics framework is tasked with managing the number of active virtual machines and servers to balance the ability of the system to respond with the energy usage. Thus Rainbow will activate resources as they are required and deactivate resources when they are no longer needed. The interval of time between probe data updates is varied between small and large intervals to examine the effect of degraded communication. These time intervals are five, ten, fifteen, thirty, and sixty minutes.

5. Results

This section presents the results of the degraded communication Rainbow framework experiments. Figure 2-6 shows the operation of Rainbow in managing the target system in increasingly DIL environments. The least intermittent environment (shown in Figure 2) has a five minute interval between communications. In this environment, Rainbow receives data and can effect required changes in a timely fashion. Figure 3 shows the environment where Rainbow's communication is limited to every ten minutes. Minor effects can be seen in the lag between when resources are needed and when they are activated. Extending to fifteen minute intervals (seen in Figure 4) causes Rainbow to not only lag in addressing resource needs, but causes its actions to overlap from day-to-day. Thus actions Rainbow takes on the first day effects how it behaves on the second day, even though the pattern of demand on the target system is the same on

day one and day two. The destabilizing effect increases as the interval between communications larger (Figures 5-6).

The DIL environments cause not only qualitative differences in Rainbow's behavior, but quantitative differences in performance. In Figure 7 the performance is measured for each of the environments by average power usage and average response time for the virtualized services per simulated time step is shown. Rainbow causes the target system to have the minimal response time, but the most power usage under than least degraded communication, the five minute intervals, having an average power usage of 2926 watt-hours and average response time of 407 milliseconds. The power usage decreases and response time increases going from to ten and fifteen minutes intervals with power usage of 2817 and 2347 watt-hours respectively and response time of 415 and 428 milliseconds respectively. Interestingly, this pattern does not continue to the thirty and sixty minute intervals. The thirty minute interval increases power usage and response time to 2778 watt-hours and the sixty minute intervals further increases both power usage and response time to 2877 watt-hours and 484 milliseconds.

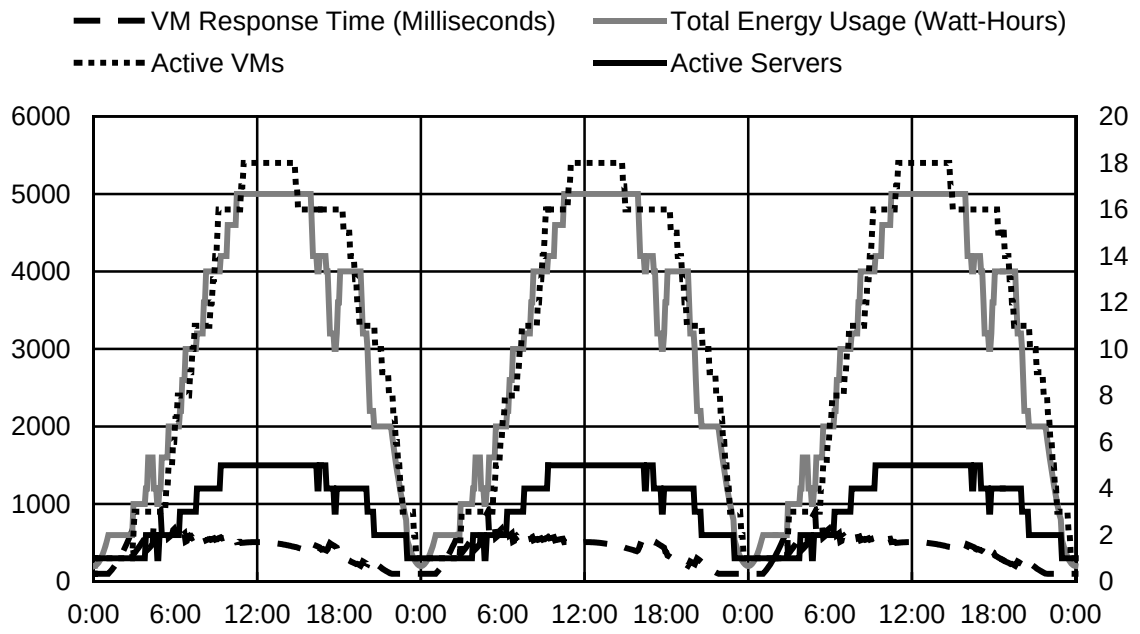

Figure 2: The performance of the Rainbow framework with five minute intervals between communications. In this environment with short intervals between communication, Rainbow can effectively activate and deactivate resources, such as virtual machines (VMs) and servers, as needed and in a timely fashion.

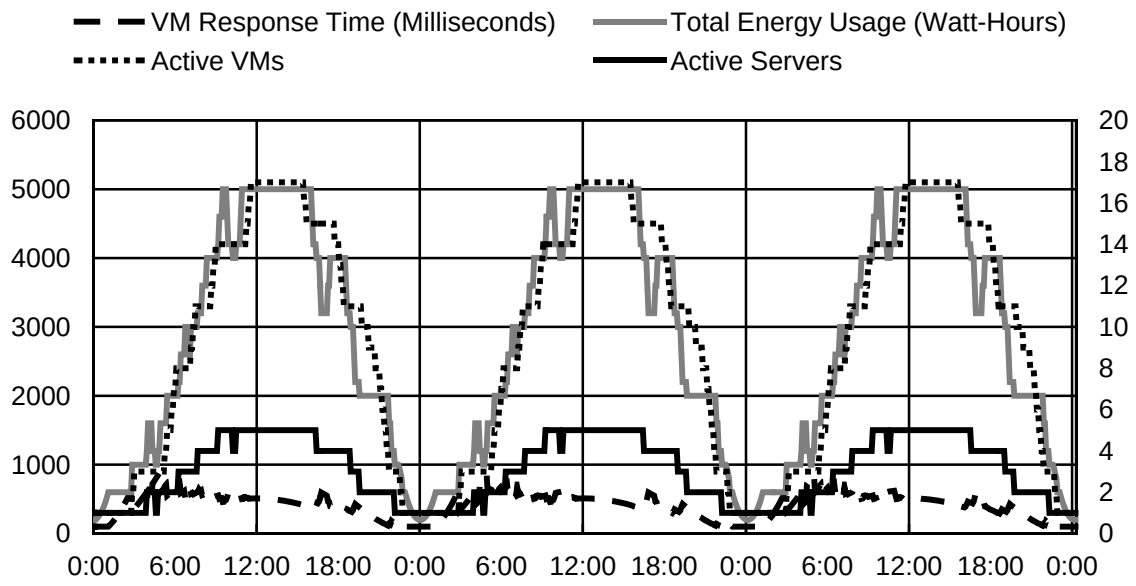

Figure 3: The performance of the Rainbow framework with ten minute intervals between communications. The slightly longer intervals between communication causes Rainbow to lag slightly behind in adding needed resources.

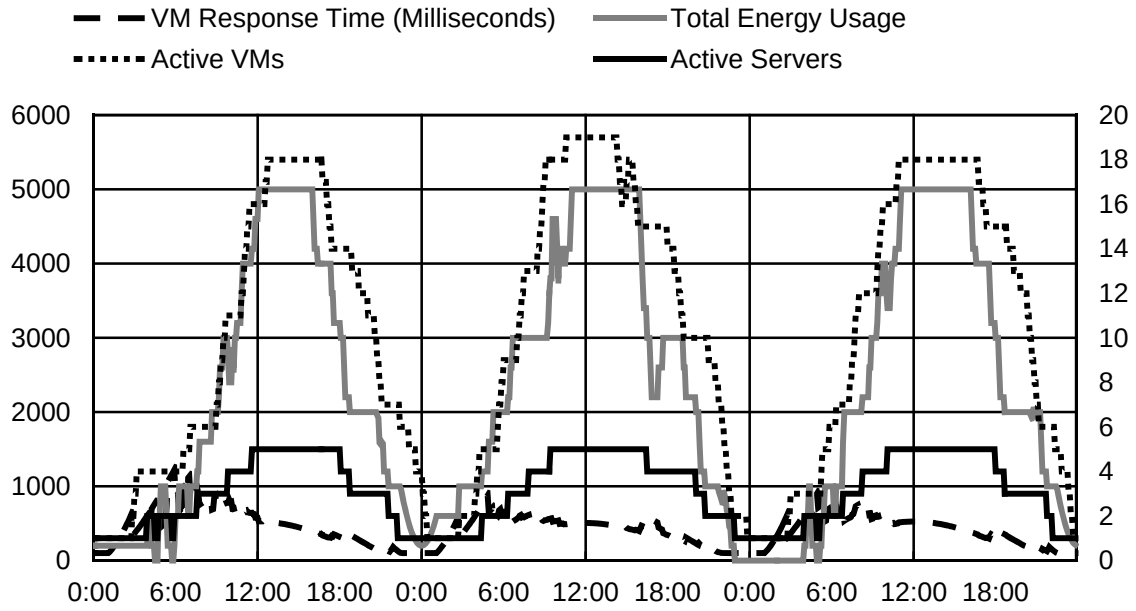

Figure 4: The performance of the Rainbow framework with fifteen minute intervals between communications. The fifteen minute intervals begin to cause an overlap between day-to-day cycles, causing variation in each day's performance.

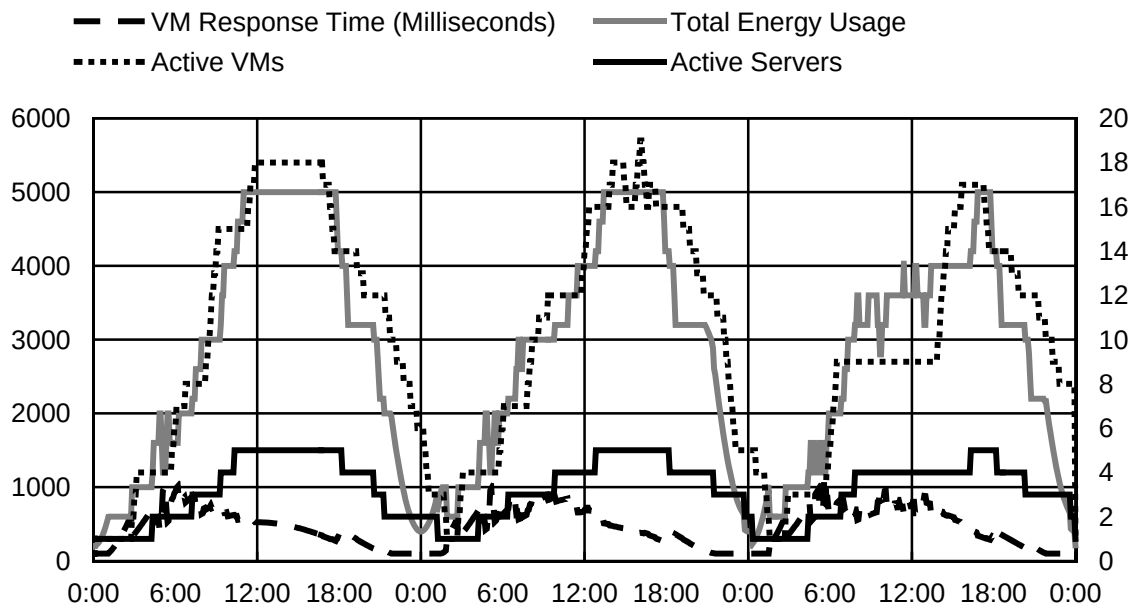

Figure 5: The performance of the Rainbow framework with thirty minute intervals between communications. Increasingly, the DIL environment causes destabilization in Rainbow's management of the target system.

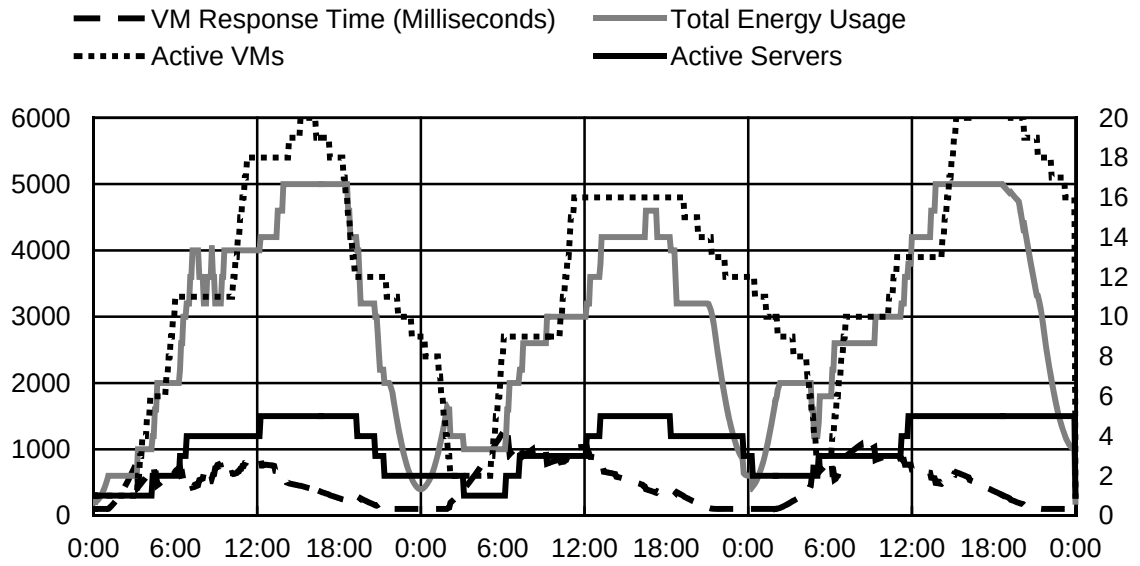

Figure 6: The performance of the Rainbow framework with sixty minute intervals between communications. The most degraded communications causes Rainbow to request resources much later than required and keep them activate longer than necessary. The overlapping interaction causes Rainbow to have too many resources online for periods of low activity.

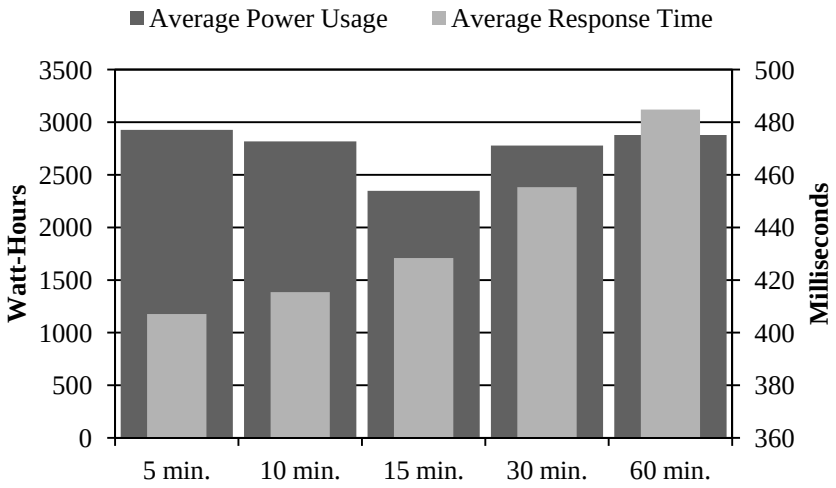

Figure 7: Average power and response time performance of the Rainbow framework across DIL conditions. Under the least degraded environment, the average power usage is 2926 watt-hours and average response time is 407 milliseconds. Increasing the intermittency to ten minutes reduces power usage to 2817 watt-hours, but increases response time to 415 milliseconds. Similarly, the fifteen minute interval decrease power usage and increases response time to 2347 watt-hours and 428 milliseconds respectively. Interestingly, the thirty minute and sixty minute intervals increase both power usage and response time, up to 2877 watt-hours and 484 milliseconds.

6. Discussion & Future Work

Autonomic computing is an increasingly critical part of cyber-physical infrastructures. This importance is due, in part, to the growing complexity of systems that requires significant labor investments for management. Therefore, systems that can manage themselves, through self-healing, self-adaptation, self-optimization, or self-defense, provide a pathway to more effective and efficient systems. However, autonomic algorithms typically rely on mechanisms that monitor the managed system and its environment to enable the self-* capabilities through detection and analysis. Disconnected, Intermittent, and Limited connectivity network environments present a challenge to autonomics, in that the ability of the system to be managed is hindered because of limited data and capability to effect changes in the system. In contrast, autonomic systems may also be the ideal approach to DIL environments because they can act quickly within the limited communication windows.

The results in this paper demonstrate that autonomic systems may suffer performance loss in DIL environments. This result is not surprising because the degrading ability to interact with the target system (both monitoring state and effecting

changes), the autonomic system's ability to heal, adapt, optimize, and defend the system similarly degrades. In particular, for centralized autonomics systems, such as the Rainbow autonomics framework, the loss of communication with target systems directly influences the ability to manage them because they are dependent on the communication to receive updates and issue commands.

One way to address this limitation is with distributed autonomics systems wherein each distributed component manages itself. In this way, no matter how disconnected the systems become, the autonomics will still be able the function. The challenge to such approaches is the limitation on globally optimizing the systems to holistically address performance degradation. Interestingly, this distributed approach may even prevent the autonomics system from solving the problem causing DIL communication because of conflicts that arise through distributed management. Alternatively, the field of machine learning (ML) can be exploited to make decisions in the brief windows of opportunity in DIL environments. Machine learning is an increasingly important component of autonomics because the size and complexity of system is beginning to outstrip the ability of humans to understand and control the maintenance, and because, the speed required for effective optimization is ever increasing[17]. In particular, one ability of machine learning to address this degradation of performance is predicting the future state of the system. Future work will focus on expanding Rainbow's capabilities through Machine Learning Gauges. Rather than simply gathering and processing data, these gauges will analyze the data to improve Rainbow's decision making ability. To address the effects of a degraded communication, ML predictors are queried to predict the future state of the system. The constraints in the modeled system are then evaluated not on the current state values, but on future state values. In this way, Rainbow can anticipate events that would require adaptation and issue commands before they are needed. This pre-emptive strategy allows autonomic frameworks to operate in degraded environments by issuing adaptive strategies in communication windows before they are needed. Such learning approaches can also be applied to make decisions based upon system state and can be used to detect anomalous operation, among other capabilities.

7. Conclusion

In summary, autonomics are playing an increasingly important role in the management of cyber-physical systems. One reason why they are becoming important is the cost of human oversight over our increasingly complex computational resources. However, a more relevant reason to DIL environments is the capability to respond faster and more optimally than a human can accomplish. This paper demonstrated that centralized autonomics approaches, such as Rainbow, can be limited by the DIL environments. This limitation is due to the inability to constantly observe and effect

changes to the system. The primary problem comes from the latency in detecting problems. Proposed future work will look at machine learning predictors that can enhance Rainbow in making decisions based upon future state of the system.

Bibliography

[1] D. Lange, P. Verbancsics, R. Gutzwiller and J. Reeder, "Command and Control of Teams of Autonomous Units," in *17th International Command and Control Research and Technology Symposium*, Fairfax, VA, 2012.

[2] D. Cohn, "Autonomic Computing," in *Proceedings of the Sixth International Symposium on Autonomous Decentralized Systems (ISAD'03)*, Pisa, Italy, 2003.

[3] A. Ganek and T. Corbi, "The Dawning of Autonomic Computing," *IBM Systems Journal*, vol. 42, no. 1, pp. 5-18, 2003.

[4] R. Murch, Autonomic Computing, Indianapolis, IN: IBM Press, 2004.

[5] D. Garlan, S.-W. Cheng, H. An-Cheng, B. Schmerl and P. Steenkiste, "Rainbow: Architecture-Based Self Adaptation with Reusable Infrastructure," *IEEE Computer*, vol. 37, no. 10, pp. 46-54, October 2004.

[6] H. Liu and M. Parashar, "DIOS++: A Framework for Rule-Based Autonomic Management of Distributed Scientific Applications," in *Proceedings of the 9th International Euro-Par Conference (Euro-Par 2003)*, Klagenfurt, Austria, 2003.

[7] D. Li and R. Muntz, "Runtime Dynamics in Collaborative," Department of Computer Science University of California, Los Angeles, CA, 1999.

[8] H. Levesque, F. Pirri and R. Reiter, "Foundations for the situation calculus," *Linköping Electronic Articles in Computer and Information Science*, vol. 3, no. 18, 1998.

[9] J. F. Allen, "Towards a general theory of action and time'.," *Artificial Intelligence*, vol. 23, pp. 123-154, 1984.

[10] M. Thielscher, "Introduction to the fluent calculus," *Linköping Electronic Articles in Computer and Information Science*, vol. 3, no. 14, 1998.

[11] D. Hirsch, P. Inverardi and U. Montanari, "Graph grammars and constraint solving for software architecture styles," in *Proceedings of the International Software*

Architecture Workshop, Orlando, FL, 1998.

[12] D. L. M´etayer, "Software architecture styles as graph grammars," in *Proceedings of the ACM SIGSOFT Symposium on Foundations of Software Engineering*, Lake Buena Vista, FL, 1996.

[13] M. Wermelinger, A. Lopes and J. L. Fiadeiro, "A graph based architectural (re)configuration language," in *Preceedings of the Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.

[14] J. Bradbury, J. Cordy, J. Dingel and M. Wermelinger, "Supporting Self-Management in Dynamic Software Architecture Specifications," in *Proceedings of WOSS'04 - ACM SIGSOFT 2004 Workshop on Self-Managed Systems*, New York, NY, 2004.

[15] B. S. a. J. A. a. D. G. a. R. K. a. H. Yan, *IEEE Transactions on Software Engineering*, vol. 32, no. 7, p. 13, 2006.

[16] B. Schmerl, A. Jonathan, D. Garlan, R. Kazman and H. Yan, "Discovering Architectures from Running Systems," *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, vol. 32, no. 7, pp. 454-465, 2006.

[17] G. Tesauro, "Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies," *IEEE Internet Computing*, vol. 11, no. 1, pp. 22-30, 2007.

[18] S.-W. C. a. D. Garlan, "Stitch: A Language for Architecture-Based Self-Adaptation," *Journal of Systems and Software, Special Issue on State of the Art in Self-Adaptive Systems*, vol. 85, no. 12, p. 38, 2012.