Adaptive Architecture in the Ballistic Missile Defense System of Systems

1 Introduction

In a System of Systems (SoS), an architecture defines how systems interact through non-physical communication [1], which impacts SoS performance. The ability to adapt between various architectures might allow for an SoS to optimize its performance for a given situation. In the Ballistic Missile Defense System (BMDS), system interaction is handle by Command and Control, Battle Management, and Communications (C2BMC) [2]. Under Maier's architecting principles, C2BMC would classify the BMDS as a directed architecture, in which individual systems are able to operate independently, but are centrally controlled by a master decision maker [1]. However, if the BMDS were to experience network outages or be the target of cyber warfare, a decentralized acknowledged architecture may prove advantageous in reaction to the new operating environment. This research aims to identify possible scenarios in which an alternate architecture would benefit BMDS performance and propose a method to optimize the BMDS architecture as a situation develops.

2 Objectives

1. Characterize the costs and risks associated with switching between architectures
2. Identify when and where adaptive architectures in the BMDS may be useful
3. Create a notional test scenario and train a SODA model to act as a surrogate predictor of system performance under different architectures
4. Identify performance thresholds in which alternative architectures are more advantageous
5. Demonstrate improved BMDS performance

3 Background

Adaptability of an SoS has largely been focused on the evolution of an SoS over time when acquiring new systems, not actively modifying the architecture in response to a changing environment. However, many of the principles applied to the evolution of an SoS over time still provide value and insight into adaptability if the BMDS is framed as an acknowledged SoS.

An acknowledged SoS is characterized as a System of Systems sharing common objectives, management, and authority, but systems still retain their own management and authority in unison with the overarching SoS goals [3–5]. There has been significant effort to characterize such systems and their evolution over time, as well as proposed methods to better optimize how such systems can negotiate and plan for optimal capability. (TODO here: Dynamic optimization, trade contracts, MUSTDO, etc). Regardless of the method, a fully realized plan for the evolution of a system requires that systems with authority acknowledge possible risks and plan for mitigating those risks [5].

In contrast to the wealth of literature on managing evolutions of an SoS on a large time scale, relatively few papers have been published investigating how an existing SoS may adapt itself to better fit a changing environment (Are there any?...). Adaptability is characterized as the ability of an SoS to react when environmental factors reduce the SoS's performance, thereby regaining some of the lost performance [6]. In contrast, evolution of an SoS takes place on long time scales with deliberate action from the SoS authorities in order to increase capability in its expected environment [].

Some system-level approaches exhibit similar problems to an SoS. For example, characterizing the performance of alternate configurations of subsystems mirrors many challenges faced by characterizing the performance of different SoS architectures. Optimization through adaptation of a system simply requires the ability to adjust component parameters [7]. Haris et. al. proposed a method for using a combination of fuzzy logic, quality function deployment, and genetic algorithms to characterize the available performance space of subsystems in relation to customer goals, risks, and budget [8]. While such approaches do show success in analyzing alternative configurations of subsystems, they fail to address critical properties that separate an SoS from a simple system. Systems-level approaches characterize a single assemblage of components which act in unison to perform a function, whereas an SoS is an assemblage of systems that share a common goal [1]. In consequence, systems-level adaptivity or optimization approaches do not address how components can be reconfigured or reorganized in order to improve performance, but rather target the modification of component parameters or the selection of a subsystem.

4 Cost and Risk

The cost and risk associated with actively changing an SoS architecture is not a trivial problem. Obvious costs and risks, such as the monetary costs of implementing a new architecture or the risk of failing to predict the future environment, are similarly encountered in system-level engineering. The more unique aspects of cost and risk in a System of Systems in outlined below.

A significant property of an SoS is that it exhibits emergent behavior, or properties that are not part of, or apparent, in individual systems [9, 10]. As such, predicting the performance of a new SoS architecture is inherently difficult and carries risk when considering implementing a new architecture. Mitigating this kind of risk, or emergent risk, requires a firm grasp on SoS dynamics, accurate models of the SoS, and a reliable prediction of the future environment.

Cost can appear in multiple forms in an adaptive SoS. The need for an adaptive architecture assumes that the SoS is not operating at full potential. Therefore, if implementing a new architecture will increase the ability of the SoS to achieve its common goal, there may be reduction in the cost of not fully meeting the common goal. The inability to adapt may also pose increased costs to an SoS. Given that an SoS will need to meet a performance target for a number of possible scenarios, some level of capability is required for each scenario. This could be met by either investing in new or upgraded systems to add to the collective, but if an adaptive approach could be used instead, a smaller set of collaborating systems may require less investment.

5 SODA Model

The SODA model acts as a surrogate to predict the performance of a BMDS architecture. The major functional elements of the BMDS are translated into nodes of the SODA model, outlined below.

Threats Threats are the root node of the system and will be modeled with a perfect self effectiveness. In the future, a threat's performance in the SODA model may be degraded to reflect a threat's ability to change course mid flight.

Sensors Sensors are only dependent on threat nodes and have a strength and criticality of dependence that reflects if a sensor is able to physically see a threat as well as the accuracy to which a threat can be identified.

Command and Control The command and control node of the SoS encompasses the ability to fuse sensor data into usable track data, asses the scenario, and make weapon assignment decisions.

Communications Communication nodes represent the communication links between physically separate nodes. For instance, a sensor will have a communication link between itself and a command and control node, but a weapons battery does not need a communication link to an interceptor because they are physically close enough to the point that the effect of communication between the two is insignificant. Network switches, which may exist between communication links, are not modeled as their contribution to the overall system performance can be effectively modeled with just the communication nodes themselves.

Interceptor Interceptors represent the ability to engage and destroy a threat. An interceptor's self effectiveness represents the probability of a successful kill given a perfect track and a single interceptor is used.

Weapons Battery Batteries are a necessary intermediate node between command and control and an interceptor in order to capture the effects of having multiple available interceptors at the system's disposal. For instance, a single interceptor with a very high self effectiveness, such as a "smart" interceptor with on-board sensing and divert capability, will perform better than a single "dumb" interceptor that lacks sensing or divert capabilities. However, if there are multiple "dumb" interceptors available, the cumulative performance of such interceptors could rival the performance of a single "smart" interceptor.

Two example architectures are given in figures 1 and 2. Physical systems, such as communication nodes and weapons batteries, persist through the architecture change from centralized to decentralized command and control. Not shown in the figures 1 and 2 are the unused communication nodes that give

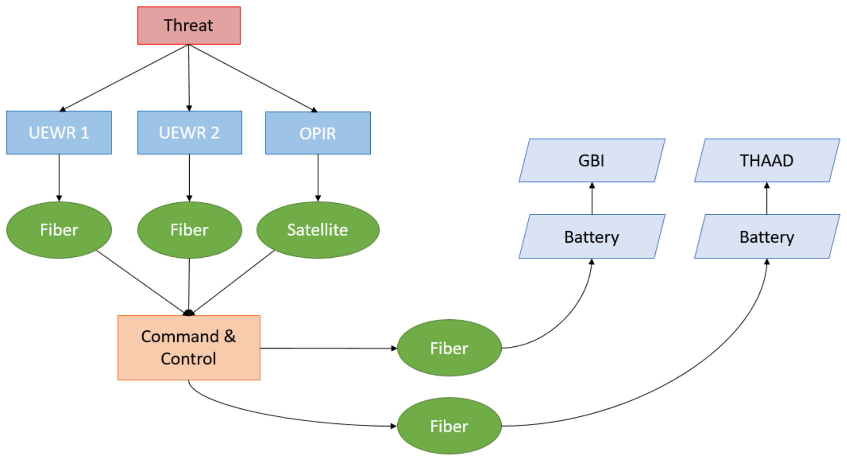

Figure 1: Centralized Architecture

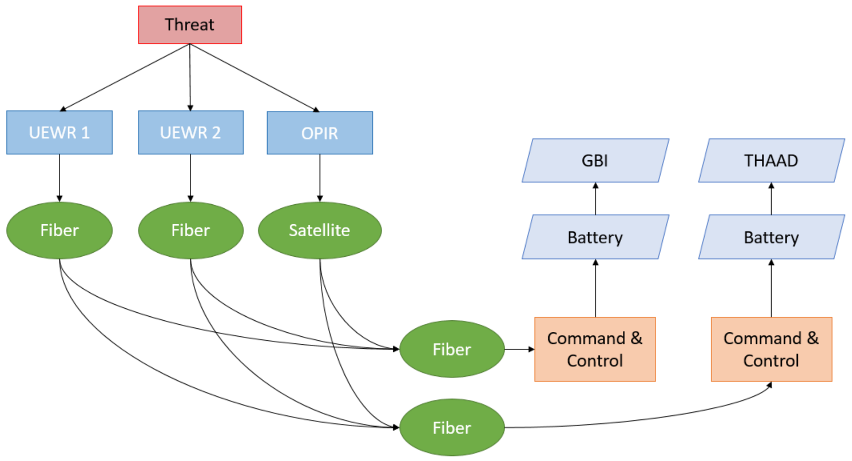

Figure 2: Decentralized Architecture

To use the SODA model as a surrogate to predict overall system performance, multiple SODA models are developed that represent the possible architectures for a given system, with node self-effectiveness values that are reflective of the current state of the agent equivalents in the agent-based model. For example, if battery has completely exhausted its supply of interceptors, its self-effectiveness will be decreased to zero.

References Cited

[1] M. W. Maier, “Architecting Principles for Systems-of-Systems,” *INCOSE International Symposium*, vol. 6, no. 1, pp. 565–573, 1996.

[2] “Command and control, battle management, and communications (c2bmc).”

[3] S. Agarwal, L. E. Pape, N. Kilicay-Ergin, and C. H. Dagli, “Multi-agent based architecture for acknowledged system of systems,” *Procedia Computer Science*, vol. 28, pp. 1–10, 2014.

[4] S. Agarwal, “Computational intelligence based complex adaptive system-of-systems architecture evolution strategy,” 2015.

[5] D. N. D. D. A. Fang, Z., “Multi-Stakeholder Dynamic Optimization for Acknowledged System-of-Systems Architecture Selection,” *IEEE Systems Journal*, 2018. Accepted for Publication 2018.

[6] R. L. Ackoff, “Towards a System of Systems Concepts,” 1971.

[7] a. de Roo, H. Sozer, and M. Aksit, “An architectural style for optimizing system qualities in adaptive embedded systems using Multi-Objective Optimization,” *Joint Working IEEE/IFIP Conference on Software Architecture, 2009 European Conference on Software Architecture. WICSA/ECSA 2009*, pp. 349–352, 2009.

[8] K. Haris and C. H. Dagli, “Adaptive reconfiguration of complex system architecture,” *Procedia Computer Science*, vol. 6, pp. 147–152, 2011.

[9] A. P. Sage and C. D. Cuppan, “On the systems engineering and management of systems of systems and federations of systems,” *Inf. Knowl. Syst. Manag.*, vol. 2, pp. 325–345, Dec. 2001.

[10] D. DeLaurentis, “Understanding Transportation as a System-of-Systems Design Problem,” *43rd AIAA Aerospace Sciences Meeting and Exhibit*, no. January, pp. 1–14, 2005.