# Deployment Optimization for Embedded Flight Avionics Systems

[1]Brian Dougherty, [2]Jules White, [1]Douglas C. Schmidt, [3]Russell Kegley and [3]Jonathan Preston
[1]Vanderbilt University, {briand,schmidt}@dre.vanderbilt.edu
[2]Virginia Tech, julesw@vt.edu
[3]Lockheed Martin Aeronautics, {russell.b.kegley,jonathan.d.preston}@lmco.com

## 1 Abstract

Loosely-coupled publish/subscribe messaging systems facilitate optimized deployment of software applications to hardware processors. Intelligent algorithms can be used to refine system deployments to reduce system cost and resource requirements, such as memory and processor utilization. This article describes how we applied a computer-assisted deployment optimization tool to reduce the required processors and network bandwidth consumption of a legacy flight avionics system.

## 2 Software Defense Application

The deployment topology of a distributed system determines how software is mapped to hardware. Optimizing the deployment topology of DoD distributed embedded systems has a significant impact on how efficiently the software utilizes the hardware. Deployment optimization can also help minimize hardware costs without requiring changes to the software or hardware architecture. This hardware reduction, in turn, helps reduce fuel consumption, increase operational ranges, and decrease cost.

## 3 Introduction

**Current trends and challenges.** Several trends are shaping the development of embedded flight avionics systems. First, there is a migration away from older *federated computing architectures* where each subsystem occupied a physically separate hardware component to *integrated computing architectures* where multiple software applications implementing different capabilities share a common set of computing platforms. Second, publish/subscribe (pub/sub)-based messaging systems are increasingly replacing the use of hard-coded cyclic executives.

These trends are yielding a number of benefits. For example, integrated computing architectures create an opportunity for system-wide optimization of *deployment topologies*, which map software components and their associated tasks to hardware processors as shown in Figure 1. Optimized deployment topologies can pack more software components onto the hardware, thereby optimizing system processor, memory, and I/O utilization [11, 13, 8]. Increasing hardware utilization can decrease the total hardware processors that are needed, lowering both implementation costs and maintenance complexity. Moreover, reducing the required hardware infrastructure has other positive side effects, such as reducing weight and power consumption.

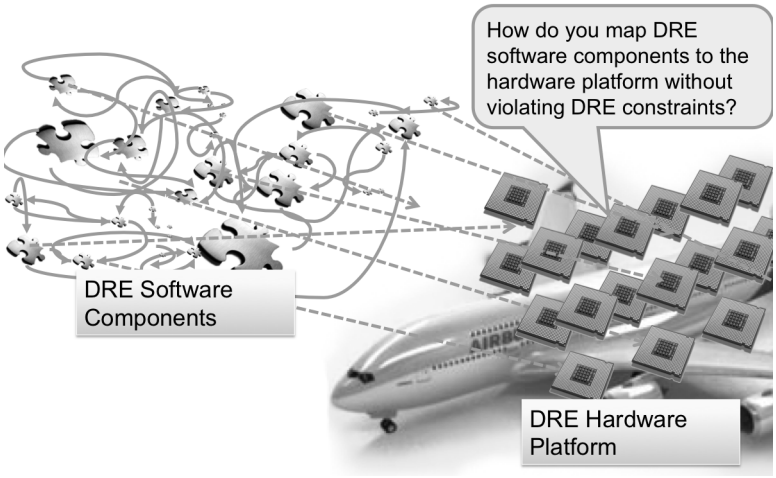


**Figure 1. Flight Avionics Deployment Topology**

**Open problems.** Developing computer-assisted methods and tools to deploy software to hardware in embedded systems is hard [1, 4] due to the number and complexity of constraints that must be addressed.

For example, developers must ensure that each software component is provided with sufficient processing time to meet any real-time scheduling constraints [12]. Likewise, resource constraints (such as total available memory on each processor) must also be respected when mapping software components to hardware components [12, 5]. Moreover, assigning real-time tasks in multiprocessor and/-or single-processor machines is *NP-Hard* [3], which means that such a large number of potential deployments exist that it would take years to investigate all possible solutions.

Current algorithmic deployment techniques are largely based on heuristic bin-packing [3, 7, 2], which represents

the software tasks as *items* that take up a set amount of space and hardware processors as *bins* that provide limited space. Bin-packing algorithms try to place all the items into as few bins as possible without exceeding the space provided by the bin in which they are placed. These algorithms use a heuristic, such as sorting the items based on sized and placing them in the first bin they fit in, to reduce the number of solutions that are considered and avoid exhaustive solution space exploration.

Conventional bin-packing deployment techniques take a one-dimensional view of deployment problems by just focusing on a single deployment concern at a time. Example concerns include resource constraints, scheduling constraints, or fault-tolerance constraints. In production flight avionics systems, however, deployments must meet combinations of these concerns simultaneously.

**Solution approach ⇒ Computer-assisted deployment optimization.** This paper describes and validates a method and tool called *ScatterD* that we developed to perform computer-assisted deployment optimization for flight avionics systems. The ScatterD model-driven engineering [10] deployment tool implements the *Scatter Deployment Algorithm* , which combines heuristic bin-packing with optimization algorithms, such as genetic algorithms [6] or particle swarm optimization techniques [9] that use evolutionary or bird flocking behavior to perform blackbox optimization. This paper shows how flight avionics system developers have used ScatterD to automate the reduction of processors and network bandwidth in complex embedded system deployments.

The remainder of this article is organized as follows: Section 4 outlines a flight avionics deployment case study we use to motivate the challenges and solutions throughout the paper; Section 5 describes the challenges faced by developers when attempting to optimize a representative flight avionics deployment topology; Section 6 discusses the ScatterD tool for deployment optimization; Section 7 provides empirical results demonstrating the reductions in hardware footprint and network bandwidth consumption that ScatterD can produce; and Section 8 presents concluding remarks.

# 4 Modern Embedded Flight Avionics Systems: A Case Study

Over the past 20 years, flight avionics systems have become increasingly sophisticated. Modern aircraft now depend heavily on software executing atop a complex embedded network for higher-level capabilities, such as more sophisticated flight control and advanced mission computing functions. To accommodate the increased amount of software required, avionics systems have moved from older federated computing architectures to integrated computing architectures that combine multiple software applications together on a single computing platform containing many software components.

The class of flight avionics system targeted by our work is a networked parallel message-passing architecture containing many computing nodes, as shown in Figure 2. At the individual node level, ARINC 653-compliant time and space partitioning separates the software applications into sets with compatible safety and security requirements. Inside a given time partition, the applications run within a hard real-time deadline scheduler that executes the applications at a variety of harmonic periods.

The integrated computing architecture shown in Figure 2 has benefits and challenges. Key benefits include better optimization of hardware resources and increased flexibility, which result in a smaller hardware footprint, lower energy use, decreased weight, and enhanced ability to add new software to the aircraft without updating the hardware. The key challenge, however, is increased system integration complexity. In particular, while the homogeneity of processors gives system designers a great deal of freedom allocating software applications to computing nodes, optimizing this allocation involves simultaneously balancing multiple competing resource demands.

For example, even if the processor demands of a pair of applications would allow them to share a platform, their respective I/O loads may be such that worst-case arrival rates would saturate the network bandwidth flowing into a single node. This problem is complicated for single-core processors used in current integrated computing architectures. Moreover, this problem is being exacerbated with the adoption and fielding of multi-core processors, where competition for shared resources expands to include internal buses, cache memory contents, and memory access bandwidth.

# 5 Deployment Optimization Challenges

This section describes the challenges facing developers when attempting to create a deployment topology for a flight avionics system. The discussion below assumes a networked parallel message-passing architecture (such as the one described in Section 4). The goal is to minimize the number of required processors and the total network bandwidth resulting from communication between software tasks.

## 5.1 Challenge 1: Satisfying Rate-monotonic Scheduling Constraints Efficiently

In real-time systems, such as the embedded flight avionics case study from Section 4, either fixed priority scheduling algorithms, such as rate-monotonic (RM) scheduling,

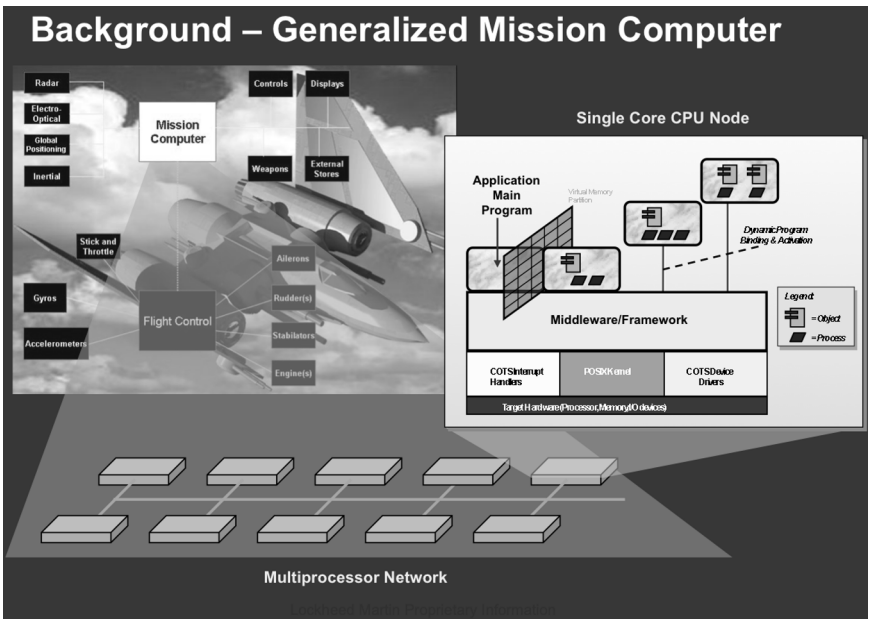


**Figure 2. An Integrated Computing Architecture for Embedded Flight Avionics**

or dynamic priority scheduling algorithms, such as earliest-deadline-first (EDF), control the execution ordering of individual tasks on the processors. The deployment topology must ensure that the set of software components allocated to each processor are schedulable and will not miss real-time deadlines. Finding a deployment topology for a series of software components that ensures schedulability of all tasks is called "multiprocessor scheduling" and is NP-Hard [3].

A variety of algorithms, such as bin-packing algorithm variations, have been created to solve the multiprocessor scheduling problem. A key limitation of applying these algorithms to optimize deployments is that bin-packing does not allow developers to specify which deployment characteristics to optimize. For example, bin-packing does not allow developers to specify an objective function based on the overall network bandwidth consumed by a deployment. We describe how ScatterD ensures schedulability in Section 6.1 and allows for complex objective functions, such as network bandwidth reduction.

### 5.2 Challenge 2: Reducing the Complexity of Memory, Cost, and Other Resource Constraints

Processor execution time is not the only type of resource that must be managed while searching for a deployment topology. Hardware nodes often have other limited but critical resources, such as main memory or core cache, necessary for the set of software components it supports to function. Developers must ensure that the components deployed to a processor do not consume more resources than are present.

If each processor does not provide a sufficient amount of resources to support all tasks on the processor, a task will not execute properly, resulting in a failure. Moreover, since each processor used by a deployment has a financial cost associated with it, developers may need to adhere to a global budget, as well as scheduling constraints. We describe how ScatterD ensures that resources constraints are satisfied in Section 6.2.

### 5.3 Challenge 3: Satisfying Complex Dynamic Network Resource and Topology Constraints

Embedded flight avionics systems must often ensure that not only processor resource limitations are adhered to, but network resources (such as bandwidth) are not overconsumed. The consumption of network resources is determined by the number of interconnected components that are not colocated on the same processor. For example, if two components are colocated on the same processor, they do not consume any network bandwidth.

Adding the consideration of network resources to deployment substantially increases the complexity of finding a software-to-hardware deployment topology mapping that meets requirements. The impact of the component's deployment on the network, however, cannot be calculated in isolation of the other components. The impact is determined by finding all other components that it communicates

with, determining if they are colocated, and then calculating the bandwidth consumed by the interactions with those that are not colocated. We describe how ScatterD helps minimize the bandwidth required by a system deployment in Section 6.3.

# 6 ScatterD: A Deployment Optimization Tool to Minimize Bandwidth and Processor Resources

Heuristic bin-packing algorithms work well for multiprocessor scheduling and resource allocation. As discussed in Section 5, however, heuristic bin-packing is not effective for optimizing designs for certain system-wide properties, such as network bandwidth consumption, and hardware/software cost. *Metaheuristic* algorithms [6, 9] are a promising approach to optimize system-wide properties that are not easily optimized with conventional bin-packing algorithms. These types of algorithms evolve a set of potential designs over a series of iterations using techniques, such as simulated evolution or bird flocking. At the end of the iterations, the best solution(s) that evolved out from the group is output as the result.

Although metaheuristic algorithms are powerful, they have historically been hard to apply to large-scale production embedded systems since they typically perform poorly on problems that are highly constrained and have few correct solutions. Applying simulated evolution and bird flocking behaviors for these types of problems tend to randomly mutate designs in ways that violate constraints. For example, using an evolutionary process to splice together two deployment topologies is likely to yield a new topology that is not real-time schedulable.

Below we explain how ScatterD integrates the ability of heuristic bin-packing algorithms to generate correct solutions to scheduling and resource constraints with the ability of metaheuristic algorithms to flexibly minimize network bandwidth and processor utilization and address the challenges in Section 5.

## 6.1 Satisfying Real-time Scheduling Constraints with ScatterD

ScatterD ensures that the numerous deployment constraints (such as the real-time schedulability constraints described in Challenge 1 from Section 5.1) are satisfied by using heuristic bin-packing to allocate software tasks to processors. Conventional bin-packing algorithms for multiprocessor scheduling are designed to take as input a series of items (*e.g.*, tasks or software components), the set of resources consumed by each item (*e.g.*, processor and memory), and the set of bins (*e.g.*, processors) and their capacities. The algorithm outputs an assignment of items to bins (*e.g.*, a mapping of software components to processors).

ScatterD ensures schedulability of the flight avionics system discussed in Section 4 by using response-time analysis. The response time resulting from allocating a software task of the avionics system to a processor is analyzed to determine if a software component can be scheduled on a given processor before allocating its associated item to a bin. If the response time is fast enough to meet the real-time deadlines of the software task, the software task can be allocated to the processor.

## 6.2 Satisfying Resource Constraints with ScatterD

To ensure that other resource constraints (such as memory requirements described in Challenge 2 from Section 5.2) of each software task are met, we specify a capacity for each bin that is defined by the amount of each computational resource provided by the corresponding processor in the avionics hardware platform. Similarly, the resource demands of each avionics software task define the resource consumption of each item. Before an item can be placed in a bin, ScatterD verifies that the total consumption of each resource utilized by the corresponding avionics software component and software components already placed on the processor does not exceed the resources provided.

## 6.3 Minimizing Network Bandwidth and Processor Utilization with ScatterD

To address deployment optimization issues (such as those raised in Challenge 3 from Section 5.3), ScatterD uses heuristic bin-packing to ensure that schedulability and resource constraints are met. If the heuristics are not altered, bin-packing will always yield the same solution for a given set of software tasks and processors. The number of processors utilized and the network bandwidth requirements will therefore not change from one execution of the bin-packing algorithm to another. In a vast deployment solution space associated with a large-scale flight avionics system, however, there may be many other deployments that substantially reduce the number of processors and network bandwidth required, while also satisfying all design constraints.

To search for avionics deployment topologies with minimal processor and bandwidth requirements—while still ensuring that other design constraints are met—ScatterD uses metaheuristic algorithms to *seed* the bin-packing algorithm. In particular, metaheuristic algorithms are used to search the deployment space and select a subset of the avionics software tasks that must be packed prior to the rest of the software tasks. By forcing an altered bin-packing order, new deployments with different bandwidth and processor

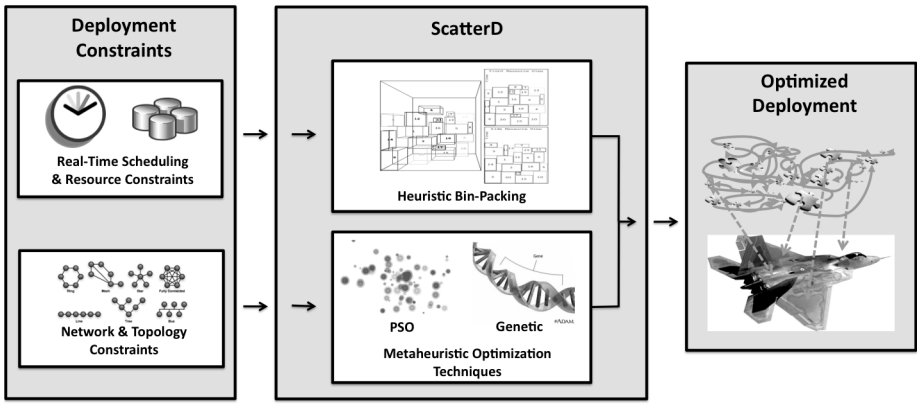


**Figure 3. ScatterD Deployment Optimization Process**

requirements are generated. Since bin-packing is still the driving force behind allocating software tasks, design constraints have a higher probability of being satisfied. By using metaheuristic algorithms to search the design space—and then using bin-packing to allocate software tasks to processors—ScatterD can generate deployments that meet all design constraints while also minimizing network bandwidth consumption and reducing the number of required processors in the avionics platform, as shown in Figure 3.

## 7 Empirical Results

This section presents the results of configuring the ScatterD tool to combine two metaheuristic algorithms (particle swarm optimization and a genetic algorithm) with bin-packing to optimize the deployment of the embedded flight avionics system described in Section 4. We applied these techniques to determine if (1) a deployment exists that increases processor utilization to the extent that legacy processors could be removed and (2) the overall network bandwidth requirements of the deployment were reduced due to colocating communicating software tasks on a common processor.

The first experiment examined applying ScatterD to minimize the number of processors in the legacy flight avionics system deployment, which originally consisted of software tasks deployed to 14 processors. Applying ScatterD with particle swarm optimization techniques and genetic algorithms resulted in increased utilization of the processors, reducing the number of processors needed to deploy the software to eight in both cases. The remaining six processors could then be removed from the deployment without affecting system performance, resulting in the 42.8% reduction shown in Figure 4.

The ScatterD tool was also applied to minimize the bandwidth consumed due to communication by software tasks

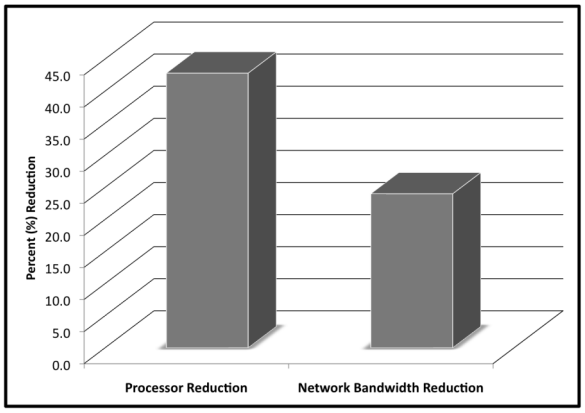


**Figure 4. Network Bandwidth and Processor Reduction in Optimized Deployment**

allocated to different processors in the legacy avionics system described in Section 4. Reducing the bandwidth requirements of the system leads to more efficient, faster communication while also reducing power consumption. The legacy deployment consumed $1.83 \cdot 10^{08}$ bytes of bandwidth. Both versions of the ScatterD tool yielded a deployment that reduced bandwidth by $4.39 \cdot 10^{07}$ or 24%, as shown in Figure 4.

## 8 Concluding Remarks

Optimizing deployment topologies on legacy embedded flight avionics system can yield substantial benefits, such as reducing hardware costs and power consumption. The following are a summary of the lessons we learned applying our ScatterD tool for deployment optimization to a legacy flight avionics system:

- **Multiple constraints make deployment planning**

**hard**. Avionics deployments must adhere to a wide range of strict constraints, such as resource, colocation, scheduling, and network bandwidth. Deployment optimization tools must account for all these constraints when determining a new deployment.

- **A Huge deployment space requires intelligent search techniques.** The vast majority of potential deployments that could be created violate one or more design constraints. Intelligent and automated techniques, such as hybrid-heuristic bin-packing, should therefore be applied to discover valid "near-optimal" deployments.

- **Substantial processor and network bandwidth reductions are possible.** Applying hybrid-heuristic bin-packing to the flight avionics system resulted in 42.8% processor reduction and 24% bandwidth reduction. Our future work is applying hybrid-heuristic bin-packing to other embedded system deployment domains, such as automobiles, multi-core processors, and tactical smartphone applications.

The ScatterD tool is available in open-source form in the Ascent Design Studio( `ascent-design-studio.googlecode.com`). A document describing the flight avionics system case study outlined in Section 4, as well as additional information on ScatterD, can be found at the SPRUCE web portal (`www.sprucecommunity.org`), which pairs open industry challenge problems with cutting-edge methods and tools from the research community.

## References


[1] H. Beitollahi and G. Deconinck. Fault-Tolerant Partitioning Scheduling Algorithms in Real-Time Multiprocessor Systems. *Pacific Rim International Symposium on Dependable Computing, IEEE*, 0:296–304, 2006.

[2] A. Bertossi, L. Mancini, and F. Rossini. Fault-Tolerant Rate-Monotonic First-Fit Scheduling in Hard-Real-Time Systems. *IEEE Transactions On Parallel and Distributed Systems*, pages 934–945, 1999.

[3] A. Burchard, J. Liebeherr, Y. Oh, and S. Son. New Strategies for Assigning Real-time Tasks to Multiprocessor Systems. *IEEE Transactions on Computers*, 44(12):1429–1442, 1995.

[4] A. Carzaniga, A. Fuggetta, S. Richard, D. Heimbigner, A. van der Hoek, A. Wolf, and COLORADO STATE UNIV FORT COLLINS DEPT OF COMPUTER SCIENCE. *A Characterization Framework for Software Deployment Technologies*. Defense Technical Information Center, 1998.

[5] W. Damm, A. Votintseva, A. Metzner, B. Josko, T. Peikenkamp, and E. Böde. Boosting Re-use of Embedded Automotive Applications Through Rich Components. *Proceedings of Foundations of Interface Technologies*, 2005, 2005.

[6] C. Fonseca, P. Fleming, et al. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the fifth international conference on genetic algorithms*, pages 416–423. Citeseer, 1993.

[7] S. Lauzac, R. Melhem, and D. Mosse. Comparison of Global and Partitioning Schemes for Scheduling Rate Monotonic Tasks on a Multiprocessor. In *10th Euromicro Workshop on Real Time Systems*, pages 188–195, 1998.

[8] L. Lehoczky, J.P. snf Sha and J. Strosnider. Enhancing Aperiodic Responsiveness in a Hard Real-Time Environment. In *Proc. of the IEEE Real-Time Systems Symposium*, pages 416–423, 1987.

[9] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, 2007.

[10] D. C. Schmidt. Model-Driven Engineering. *IEEE Computer*, 39(2):25–31, 2006.

[11] L. Sha and J. Goodenough. Real-time scheduling theory and Ada. *Computer*, 23(4):53–62, 1990.

[12] J. Stankovic. Strategic Directions in Real-time and Embedded Systems. *ACM Computing Surveys (CSUR)*, 28(4):751–763, 1996.

[13] J. Strosnider and T. Marchok. Responsive, deterministic IEEE 802.5 token ring scheduling. *Real-Time Systems*, 1(2):133–158, 1989.