# Proactive and Reactive Resource Allocation

Joseph K. Cross

Lockheed Martin Tactical Systems
P.O. Box 64525, M.S. U2X26
St. Paul, MN 55164-0525
(651) 456-7316
joseph.k.cross@lmco.com

Patrick J. Lardieri

Lockheed Martin
Advanced Technology Laboratories
1 Federal Street, A&E 3W
Camden, NJ 08102
patrick.j.lardieri@lmco.com

## Abstract

*Resources, such as processors and network bandwidth, are allocated to application and common services. In long-running systems, it is common to re-allocate these resources, due to changing mission requirements or to changing resource availability. Two patterns of resource allocation are described in this paper: reactive resource allocation, which begins when the need for reallocation arises, and proactive resource allocation, which is planned before the need arises.*

*Reactive resource allocation is widely used in both commercial and United States Department of Defense (DoD) distributed real-time and embedded (DRE) systems. In addition, reactive resource allocation is the focus of a vigorous research and development community. Proactive resource allocation, on the other hand, seems to be employed primarily, if not exclusively, in DRE systems, where in some cases it provides the only means to meet real-time response and recovery requirements.*

## Reactive Resource Allocation

---

The Reactive Resource Allocation design pattern maintains the performance of a computing system within required bounds by observing when the system's performance is close to crossing, or has crossed, a threshold; determining how best to reallocate the available resources to prevent or correct any requirement violation; and then directing that resource reallocation.

---

### Example

The combat system on a warship contains an object called a *tracker*. The tracker serves as a real-time database of tracks, where a *track* is an object that represents a physical, moving object that is of tactical interest to the ship, such as a friendly or hostile aircraft, missile, ship, or submarine. The tracker accepts inputs from sensors such as radars, and uses these inputs to update its tracks. The tracker's task is computationally intensive, and its performance is subject to severe real-time constraints. The number of CPU cycles per second that the tracker requires depends on the number of tracks in its database and the number of sensor reports per time interval.

A performance monitor object continuously observes the behavior of the tracker. The monitored quantities may be direct measures of its performance, such as the time required for the tracker to respond to queries, or indirect measures, such as the number of tracks currently in the tracker's database.

When the performance monitor notes that a measure of performance is approaching a pre-defined limit, the performance monitor notifies a *resource allocator*. The resource allocator examines the currently available resources and reallocates them appropriately. For example, the resource allocator may split the tracker object into several sub-trackers, where each sub-tracker executes on a different processor, and each sub-tracker maintains only a subset of the entire set of active tracks. Interactions with the (now virtual) tracker object that are initiated by other objects are routed to the appropriate sub-tracker by a servant locator [2] that implements the Interceptor pattern [3].

An excellent example of reactive resource reallocation technology is provided by the DARPA Quorum project [4].

## Context

- A system with quality of service (QoS) constraints that is subject to changing service requirements and/or changing availability of resources.
- The system must be resource-constrained, as may be limited by physics or economics, to the extent that some resources must be used for different purposes at different times.
- The system must be able to tolerate out-of-spec operation for a period of time (possibly unbounded) as the system adapts to the requirement and/or resource change.
- Neither the exact time nor circumstances to which the system will have to adapt to can be fully specified in advance.
- Determining a *best* reallocation requires information about the configuration of the system at the time the adaptation must occur.

## Problem

In long-lived, complex, computational systems, no single allocation of resources to services is likely to be permanently optimal, or even acceptably efficient. Hence resource allocation and reallocation will be required during system execution. The problem of reallocation involves determining that the system is failing to meet one or more critical performance requirements; adapting either the application load or resource assignment to improve the system's performance; and accomplishing the reallocation without disrupting the system operation for an excessive period of time. Accomplishing these requires resolving the following forces:

- The monitoring of system performance must impose a minimal overhead on the normal operation of the system; in particular, the benefit of reallocating resources must exceed the cost of monitoring.

- The triggering and allocation determination functions must be either predictable or fast, and the reallocation function must be fast, in order to minimize the period of time during which system QoS constraints are violated.
- The allocation determination function should be flexible, to allow for resource failures and for the installation of new resources.
- The allocation determination function can be simplified and accelerated by restricting certain resources to support only specified services ("stovepiping"), but resource utilization will likely be thereby decreased.
- The combination of the triggering function with the allocation determination function must be stable, to ensure that after reallocation another reallocation will not be immediately initiated, which could lead to system collapse.
- The allocation determination function must respect the criticalities of current allocations, such as TCP connections that must not be broken, or messages must not be lost.
- The reallocation function must respect criticalities of on-going services (e.g., avoiding dropped messages).

## Solution

Many known solutions, for example DeSiDeRaTa [4], implement the monitoring function using multiple objects. Each object monitors one or more aspects of the system behavior. The monitored quantities may be direct measures of system performance (i.e. QoS values) or indirect measures (i.e., quantities that have a known relationship with qualities of service).

The triggering function is an object that implements the Observer pattern [1] to receive notification when a subset of the monitor objects reports a significant change (determined by the triggering function) in a monitored quantity. When such a significant change is detected, the triggering function recommends a reallocation be performed. Implementing the triggering function using the Strategy pattern[1] enables the triggering decision logic to be changed easily at design or run time. If there are multiple orthogonal performance requirements consider implementing multiple triggering functions. If there is strong interdependence between multiple performance requirements then consider implementing a higher level triggering coordinator that integrates the inputs from the lower level triggering functions.

The allocation determination function implements the Observer pattern [1] to be notified when any of the triggering functions recommends a reallocation. The allocation determination function can choose to delay or ignore the reallocation recommendations. For example, there may be lock out time when the system is forbidden from reallocating. The allocation determination function typically requires a model of the current system configuration to reason about reallocations. The allocation determination function implements analytical and/or heuristic logic to predict whether a proposed reallocation will actually improve system performance. Finally the allocation determination function should also be implemented using the Strategy pattern [1] for the same reasons stated above.

Once a new allocation is developed individual QoS controls in the infrastructure and application must be reconfigured. The allocation configuration function implements the new allocation by directing these controls. It receives reallocation direction from the allocation determination function  by implementing the Observer pattern [1].

## Consequences

Reactive Resource Reallocation provides the following benefits:

- The ability to gracefully maintain end-to-end qualities of service within specified bounds over a wide variety of load and failure conditions.
- The ability to make effective use of reactive capabilities that are built into COTS components, such as routers.

However, the Reactive Resource Reallocation pattern encounters the following liability:

- A long time may be required to react to a sudden change in service requirements or infrastructure capabilities if the change invalidates a large proportion of the current resource allocations, if the infrastructure is complex, or if the infrastructure is geographically dispersed.

# Proactive Resource Allocation

The Proactive Resource Allocation design pattern maintains the performance of a computing system within required bounds by anticipating critical changes in the system state, pre-planning resource allocations appropriate to those changes, monitoring the system state for those changes, and implementing the appropriate pre-planned resource allocation when the system state changes.

## Example

In order to improve the quality of life for personnel on board future warships, plans call for the distribution of crew entertainment video over the ship's backbone communication infrastructure.

When such a ship transitions from its normal mode of operation to battle mode, as might be triggered by the detection of an incoming missile, the distribution of such entertainment video must be immediately terminated and the communication bandwidth it occupied must be re-allocated to more critical functions. Since there will not be time for protracted negotiations among critical functions for access to the communication resources, such access must be pre-planned.

## Context

- A system with quality of service (QoS) constraints that is subject to changing service requirements and/or changing availability of resources.
- The system must be resource-constrained, as may be required by physics or economics, to the extent that some resources must be used for different purposes at different times.
- The system can only tolerate out-of-spec operation for a bounded (and typically short) period of time as it adapts to the requirements and/or resource change.
- A set of exact times or circumstances to which the system will have to adapt to are selectable in advance.
- Determining an *acceptable* reallocation can be performed in advance of the time of the reallocation.

## Problem

In long-lived, complex, computational systems, no single allocation of resources to services is likely to be permanently optimal, or even acceptably efficient. Hence resource allocation and reallocation will be required during system execution. However, in safety or mission critical systems timing constraints must be met to ensure the system behaves properly. This property typically eliminates dynamic adaptive reallocation (as defined in Reactive Resource Allocation) as a potential solution. Instead, specific critical "reallocation points" are identified in advance – these may include faults, mission

changes, resources changes, system events, environmental changes. New system allocations are computed for each of these "reallocation points" prior to system operation. The computation of the solution is typically done during design or online as part of transitioning from one system allocation to another. In the latter case, the computation must be bounded in execution and must produce provably acceptable solutions. Accomplishing these requires resolving the following forces:

- The monitoring of system performance must impose a minimal overhead on the normal operation of the system; in particular, the benefit of reallocating resources must exceed the cost of monitoring.
- The allocation determination function must produce reallocations that are provably acceptable (meaning that they are correct and provide and acceptable level of performance).
- The worst case execution time of the triggering and reallocation functions must be predictable and typically fast in order to minimize the period of time during which system QoS constraints are violated.

## Solution

Divide the (possibly very large) state space of the system into subsets called *modes*. (Abstractly, a mode may be regarded as a Boolean-valued function on the system state space; 'battle mode' is a mode, and 'at least one engine is operational' is a mode.) Express QoS and functional requirements as functions of mode; e.g., "When at least one engine is operational, engine RPM values shall be distributed with a latency of at most 50 milliseconds."

Many known solutions, for example DeSiDeRaTa [4], implement the monitoring function using multiple objects. Each object monitors one or more aspects of the system behavior. The monitored quantities may be direct measures of system performance (i.e. QoS values) or indirect measures (i.e., quantities that have a known relationship with qualities of service).

The triggering function is an object that implements the Observer pattern [1] to receive notification when a subset of the "monitor objects" reports a significant change (determined by the triggering function) in a monitored quantity. The triggering function determines when conditions warrant a mode change. Multiple triggering objects may be used to address one or more mode changes. Implementing the triggering function using the Strategy pattern[1] enables the triggering decision logic to be changed easily at design or run time

The allocation determination function is responsible to specify a reallocation for each system mode in advance of a mode change. The allocation determination function maybe part of the system design process, a part of the management software that conducts mode changes, or both. In any case it is critically important that the worst case execution time to complete a mode change be bounded (and typically fast).

Implementing the allocation determination function as part of the system design process is typically done via a combination of techniques. Analytical techniques (e.g. scheduability analysis) are used to predict system performance in specific configurations.

Systems simulation is used to further understand system behaviors that can arise dynamically. And system prototyping is used to verify expected results obtained from analysis and simulation. At the end a well defined system configuration is defined for each mode. Upon a mode change the system implements the configuration as specified.

Implementing the allocation determination function as part of the management software that conducts mode changes is typically done using very conservative techniques. In order to determine a new allocation in bounded time—well understood and well behaved algorithms are used to compute new allocations. This is usually suitable when the problem lends itself to a concise mathematical description (e.g. like a closed loop control system) and is very difficult otherwise.

The "online" allocation determination function must understand the state of the system to generate a new allocation. Partial state knowledge may be derivable from the current and transitioning mode. Additional state knowledge may be built up by the allocation determination function, preferably prior to the mode change event. The allocation determination function can construct a system model by subscribing to relevant state monitors.

Proactive Resource Allocation is typically combined with Reactive Resource Allocation. Proactive is used for the absolute mission and safety critical aspects of the systems and reactive is used for the remainder. In these hybrid approaches the Reactive Resource Allocator operates within a resources envelope defined by the Proactive Resource Allocator. In this way the Proactive Allocator insures that the Reactive Resource Allocator cannot interfere with critical system functions.

### Consequences

Proactive Resource Reallocation provides the following benefits:

- The ability to provide specified end-to-end qualities of service quickly following an anticipated change in system state.
- The ability to determine that specified qualities of service cannot be provided in advance of their need (and thereby to prevent a problem rather than responding to the problem).

However, the Proactive Resource Reallocation pattern encounters the following liabilities:

- Not every mode change can be anticipated (e.g., multiple resource failures); hence support for some reactive resource reallocation is generally desirable.

## Acknowledgment

This work was sponsored by DARPA under contract number F33615-01-C-1847.

## References

[1] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns, Addison-Wesley, 1995

[2] N. Wang, K. Parameswaran, and D. C. Schmidt, “The Design and Performance of Meta-Programming Mechanisms for Object Request Broker Middleware,” in *Proceedings of the 6th Conference on Object Oriented Technologies and Systems* (San Antonio, TX), USENIX, Jan/Feb 2000

[3] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, Pattern-Oriented Software Architecture—Patterns for Concurrent and Distributed Objects, John Wiley and Sons, 2000

[4] Quorum http://www.darpa.mil/ito/research/quorum/index.html