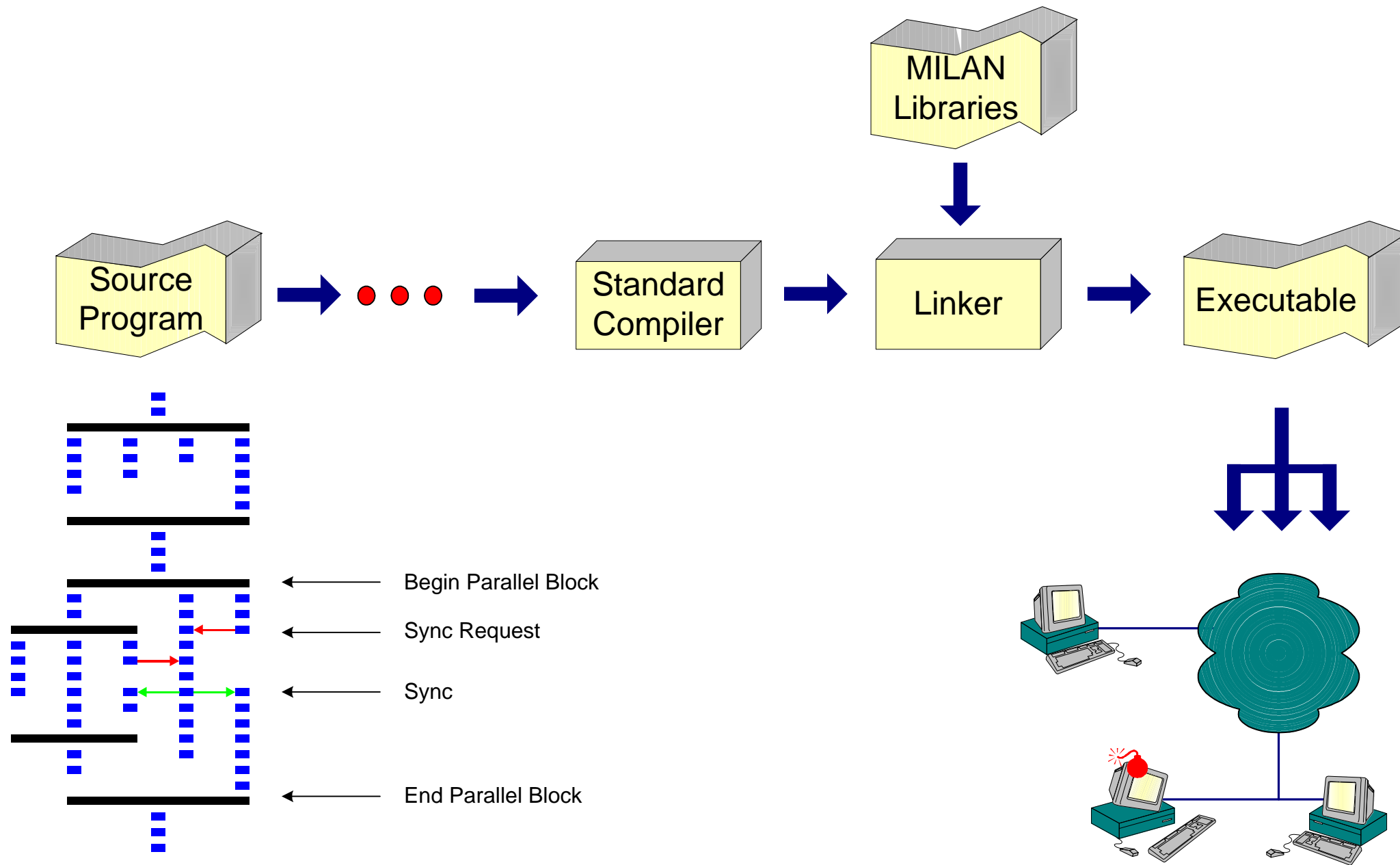
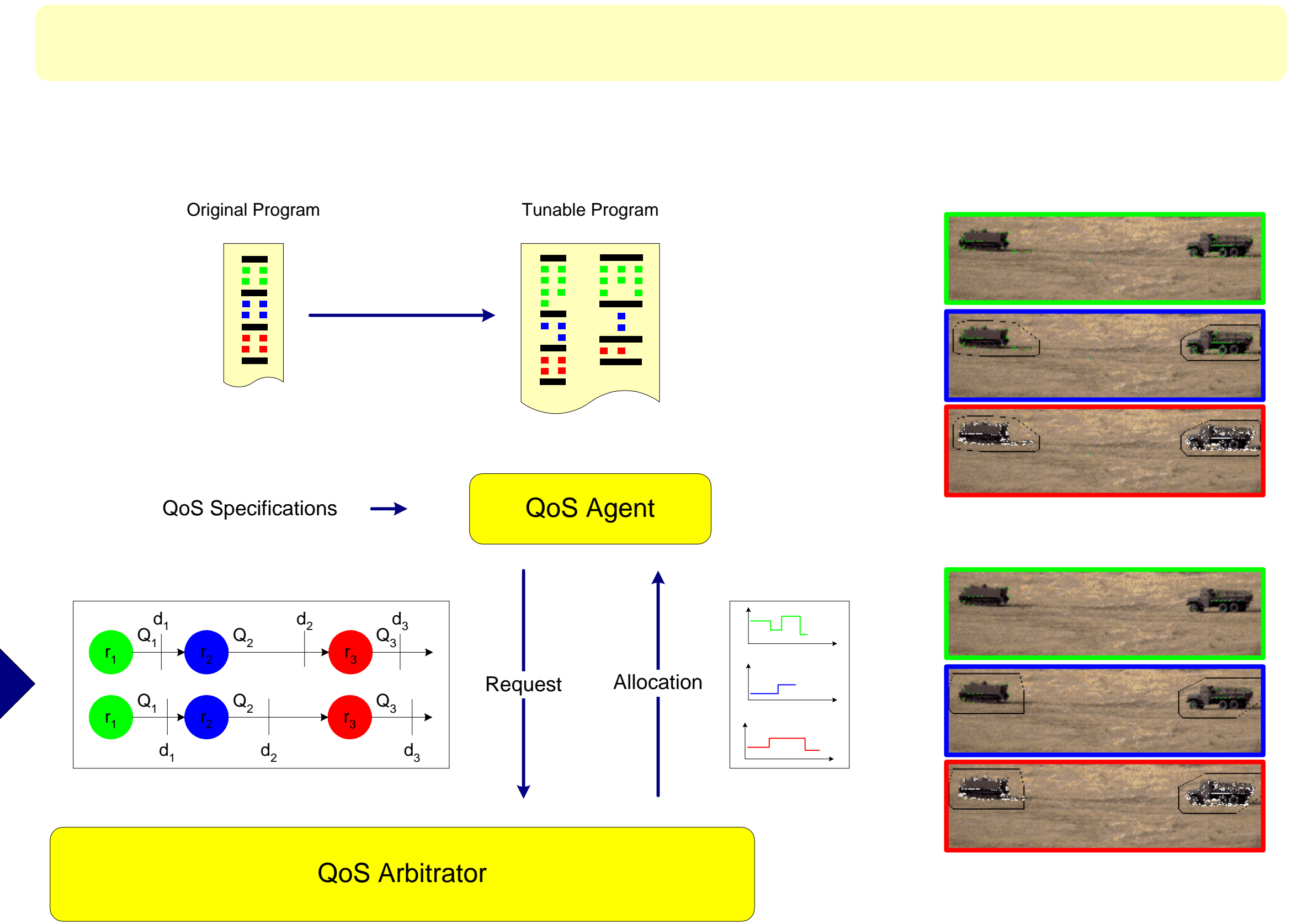
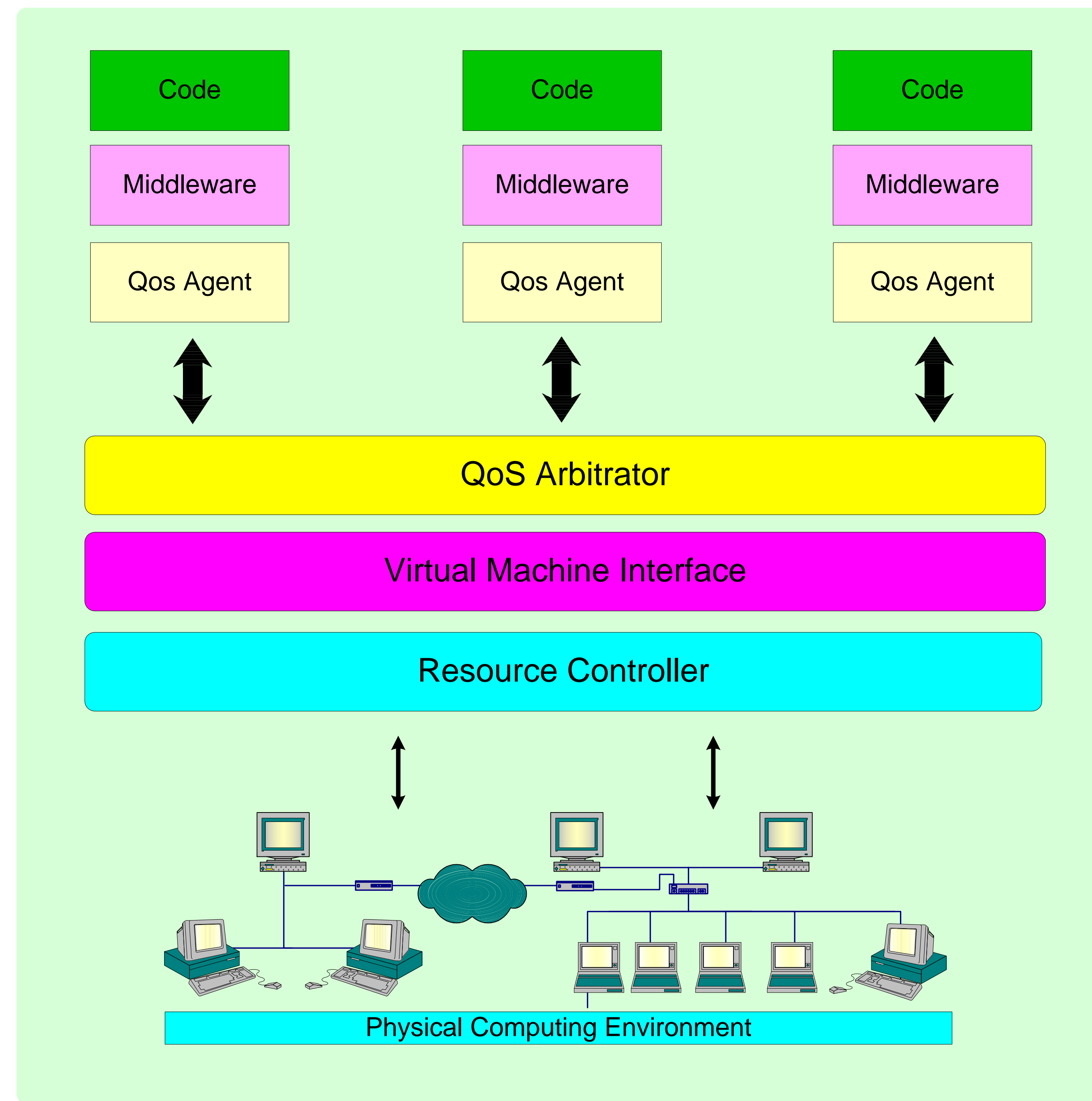


Metacomputing In Large Asynchronous Networks

Calypso, Chime, Charlotte: Fault-tolerant Execution

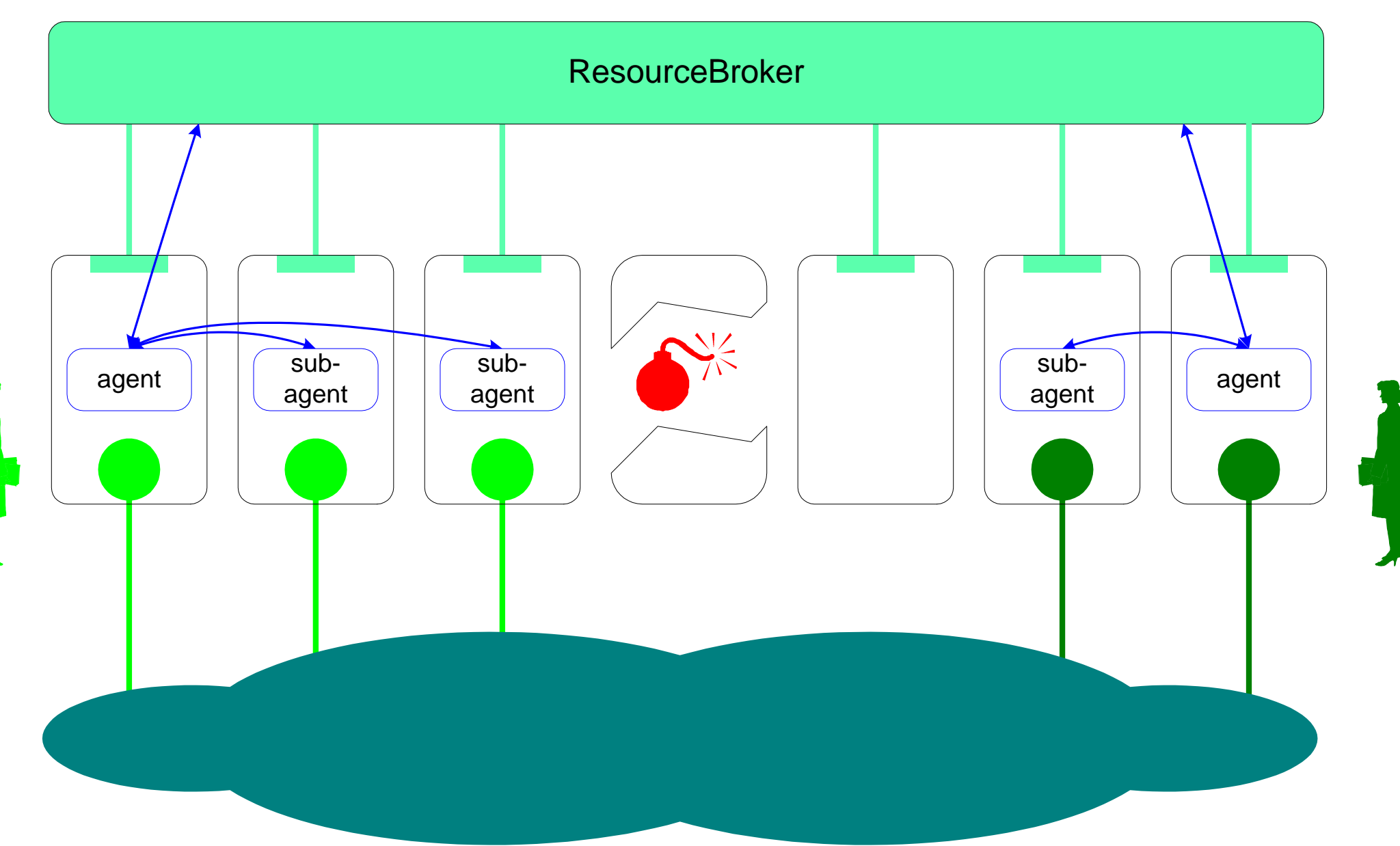


Calypso, Chime, and Charlotte provide programming and runtime support for adaptive computations on distributed platforms. The source program targets a **perfect virtual shared-memory machine**. The executable contains embedded code for distribution, scheduling, memory management, load balancing, and fault masking. The computation **transparently adapts** to the dynamically changing environment.



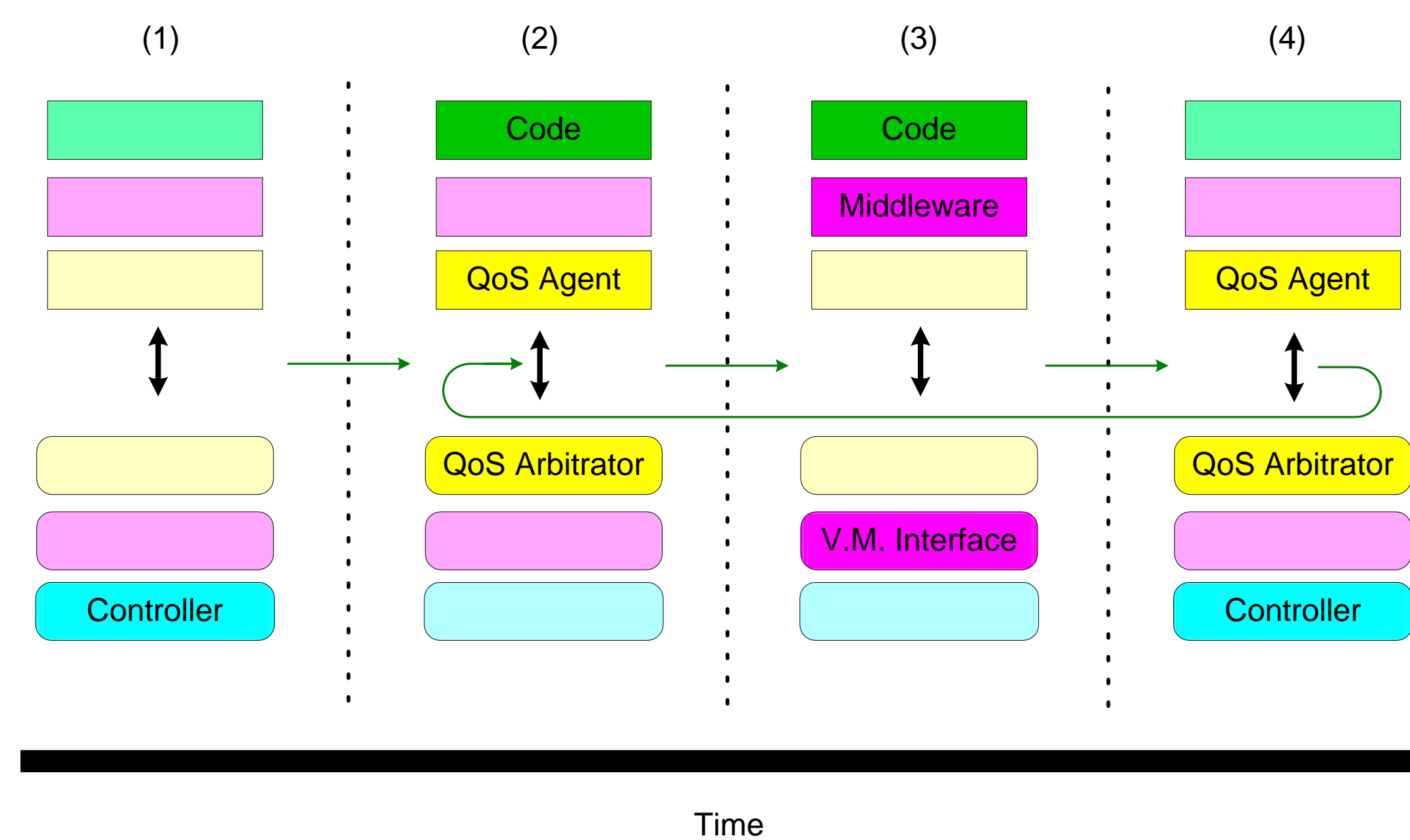
QoS Arbitrator
adaptive computations in a distributed system. The system-level **QoS arbitrator** takes advantage of a flexible program specification, transmitted by the application **QoS agent**, exploiting computation tunability to trade off resource allocation during the program's lifetime, and computation adaptability to preemptively allocate, deallocate, and reallocate resources as required.

ResourceBroker: A Responsive Resource Controller

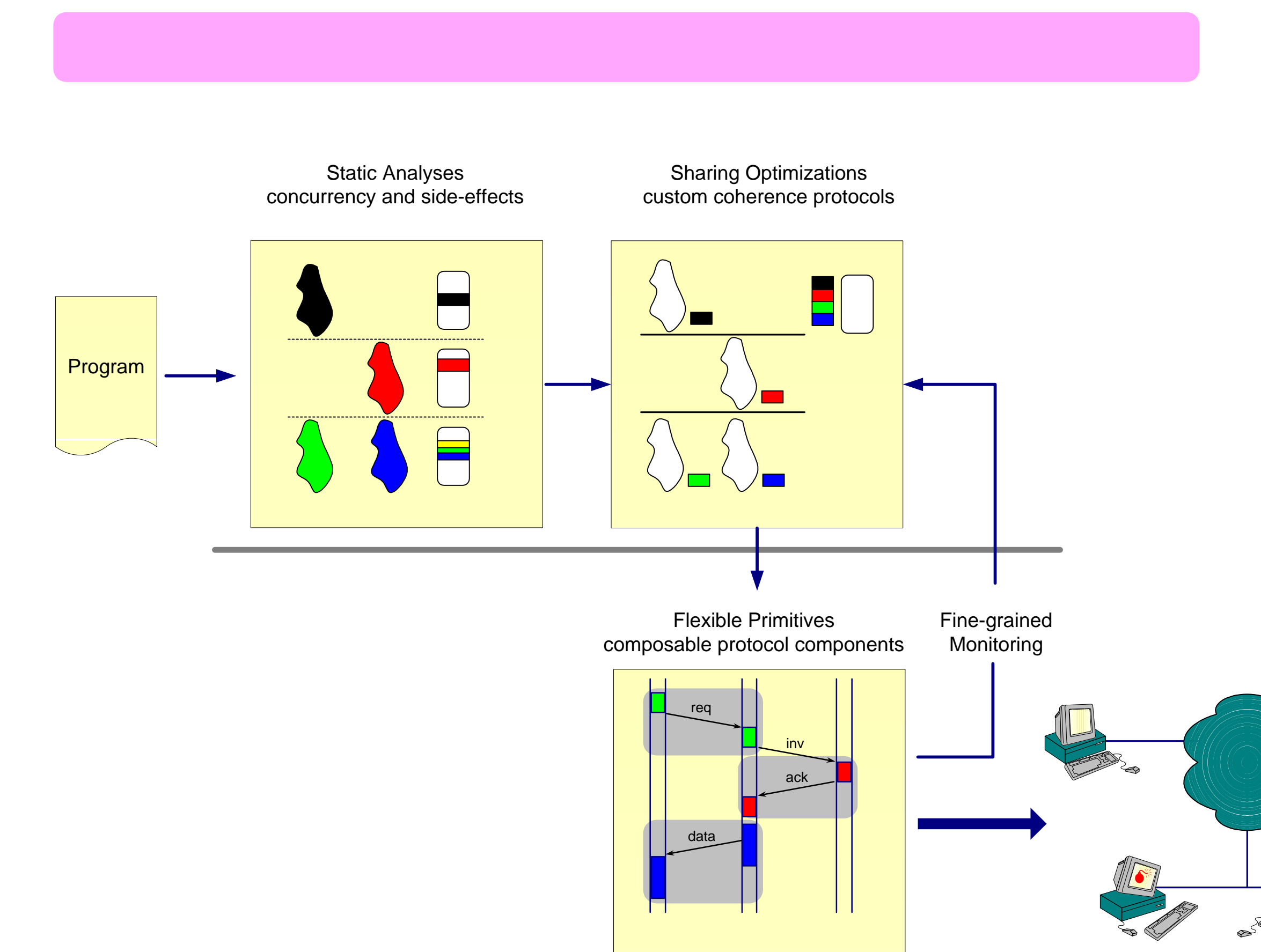


ResourceBroker transparently manages **multiple virtual machines** for COTS hardware, systems software, and diverse programming environments. It creates and monitors the virtual machines, restructuring and rescaling them dynamically to reflect the computations' needs and the changing resource availability.

The MILAN software architecture consists of three application-level layers interacting with three system-level layers. The application layers provide (from top to bottom) a platform-independent **specification** of the computation, **application-aware middleware** which enables the computation to adapt to changing resource availability profiles, and a **QoS agent** which negotiates an appropriate level of resource allocation during the program's lifetime. The system layers provide (from bottom to top) a **resource controller** which keeps track of resource availability, an application-independent **virtual-machine interface** layer that enables user computations to execute across heterogeneous resources in a fault-tolerant and efficient fashion, and a **QoS arbitrator** which responds to QoS agent requests with a level of resource allocation that can satisfy application QoS requirements.



Different layers of the MILAN architecture are active at different times in the program's lifetime. The above figure shows a typical progression: (1) the resource controller layer adds resources into the system, (2) on program arrival, the user program, the application QoS agent, and the system QoS arbitrator participate to determine a resource allocation for the program, (3) the user program and application-aware middleware stages adapt to the available resources, (4) a change of resource availability is detected by the resource monitor which triggers a renegotiation of resources between the application and the system (step 2).



Access Proxies
DSM systems via protocol customization. The framework first builds a model of program interaction using **compile-time analyses** and **run-time profiling**, and then, utilizing this model, derives code fragments (access proxies) which construct **custom protocols** from **flexible run-time primitives**.