Hybrid Resource Management for Dynamic Real-Time Systems

Lonnie R. Welch, Purvi Shah, Shikha Jain

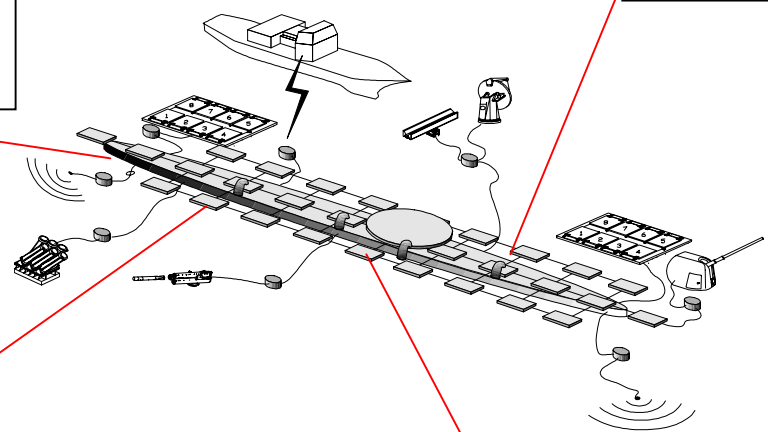
Laboratory for Intelligent Real-Time Secure Systems
School of EECS
Ohio University

http://zen.ece.ohiou.edu/

Distributed mission-critical real-time systems

Timely response to threats

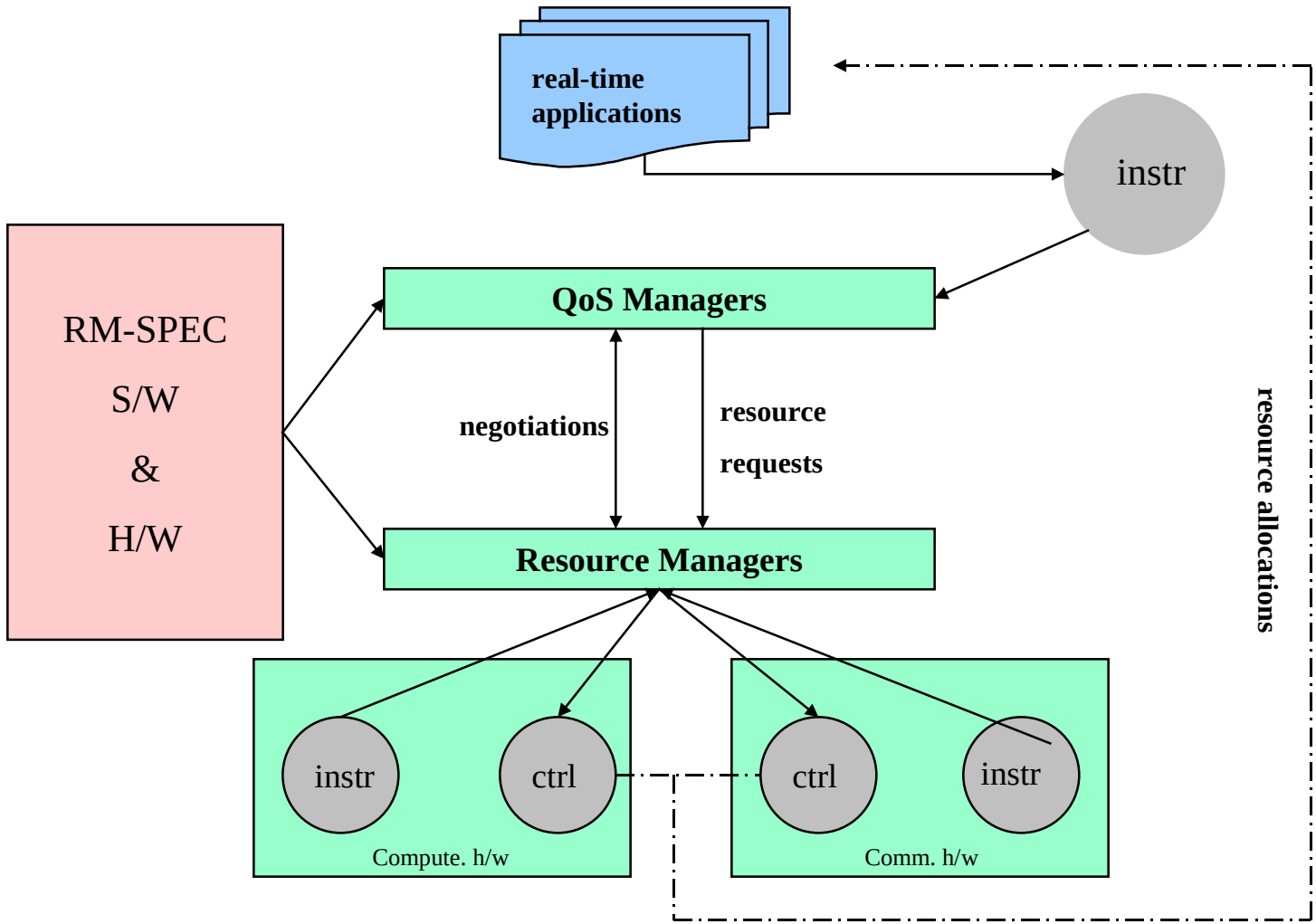
Dependable behavior

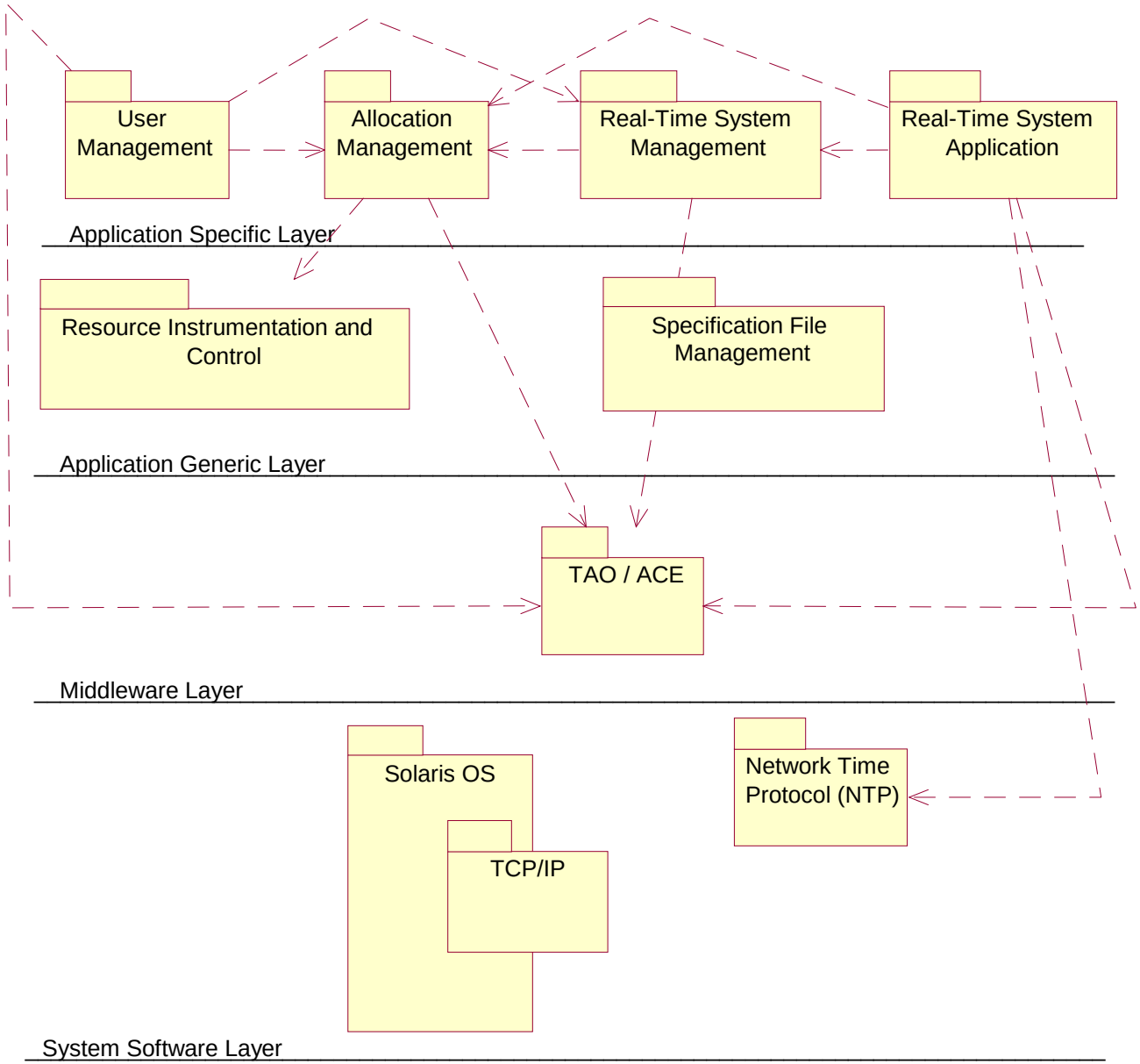


Efficient use of resources

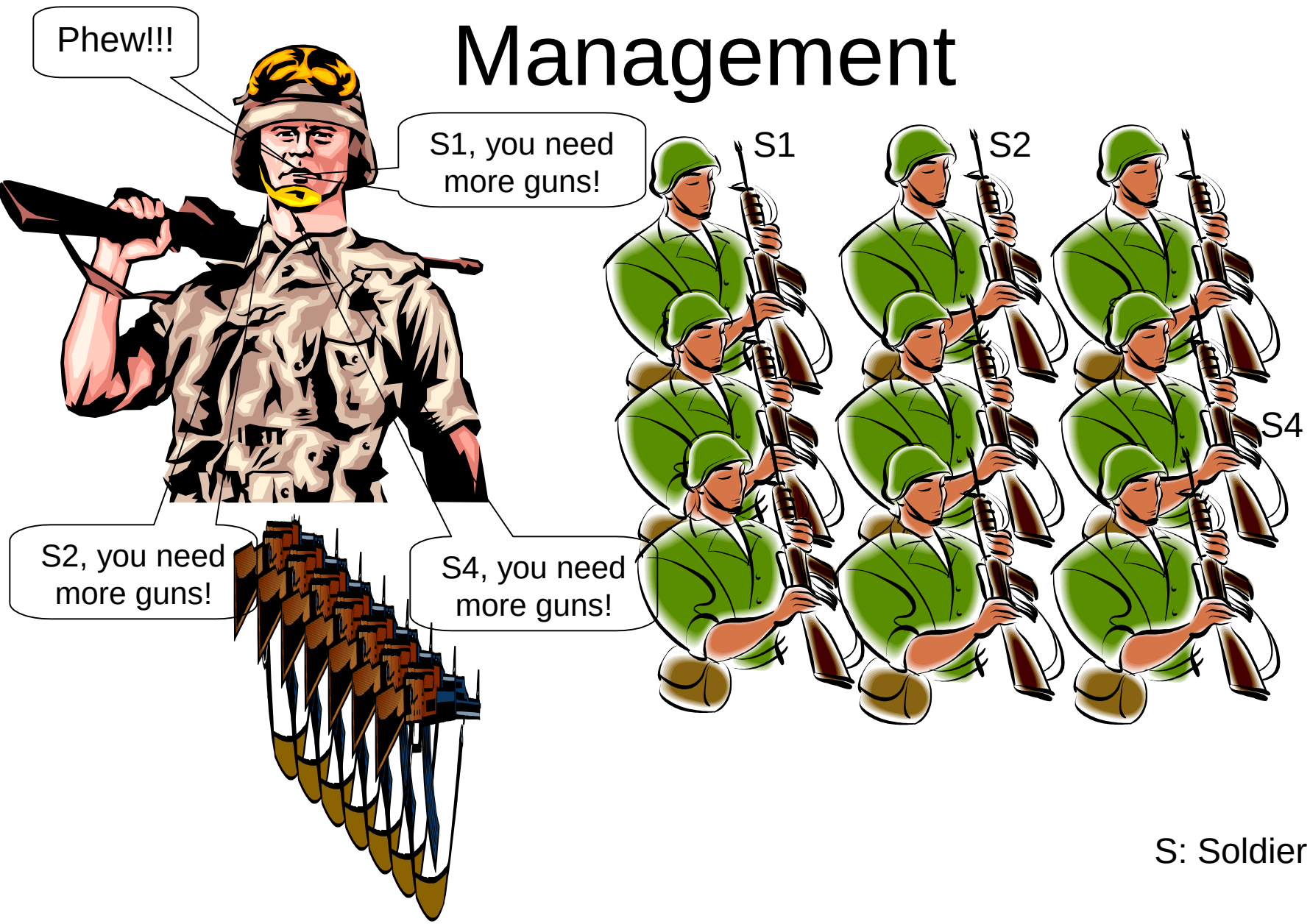
Scalability

Resource Management Architecture



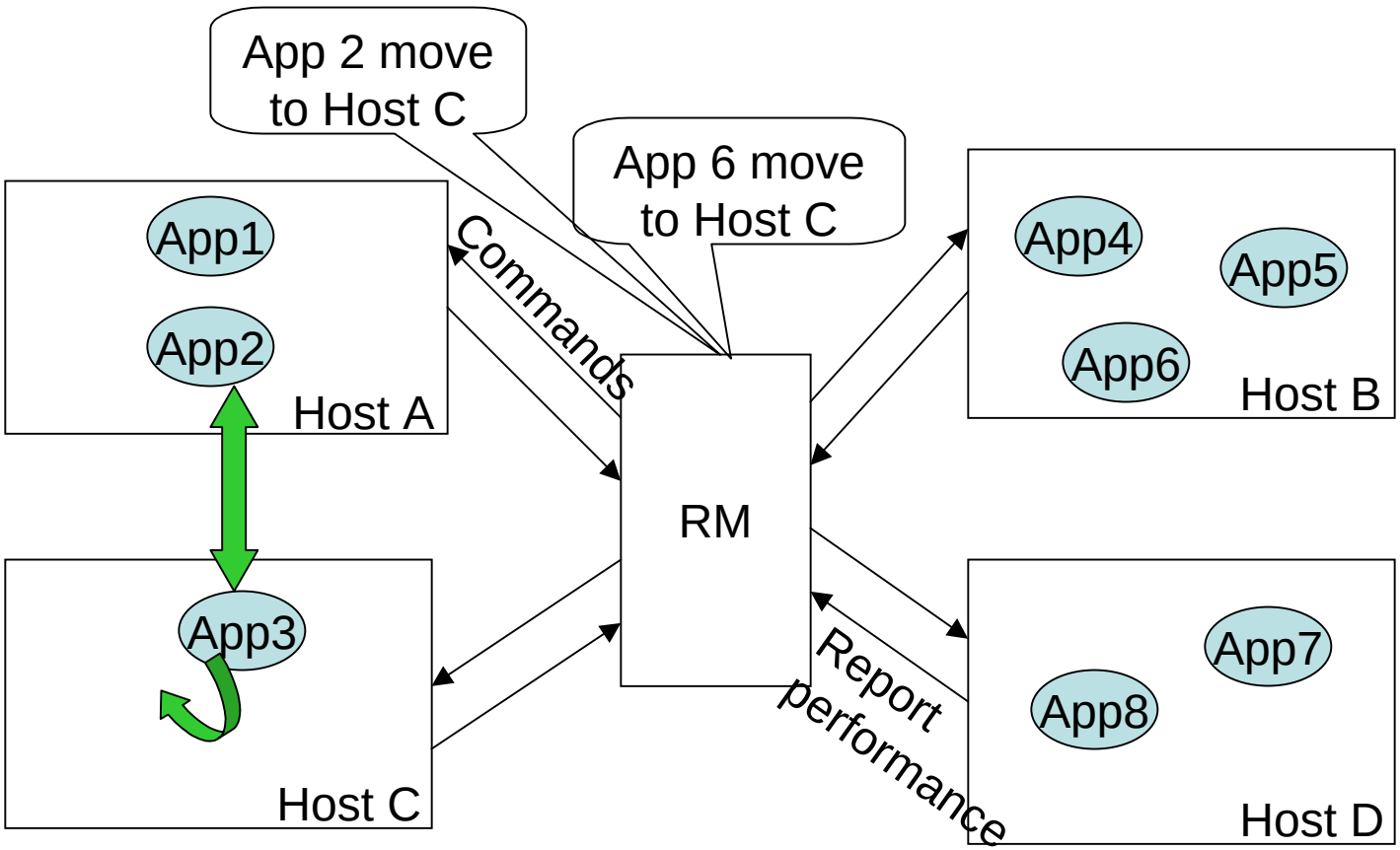

User Management
Allocation Management
Real-Time System Management
Real-Time System Application
Application Specific Layer
Resource Instrumentation and Control
Specification File Management
Application Generic Layer
TAO / ACE
Middleware Layer
Solaris OS
TCP/IP
Network Time Protocol (NTP)
System Software Layer

Centralized Resource Management



S: Soldier

Centralized Resource Management



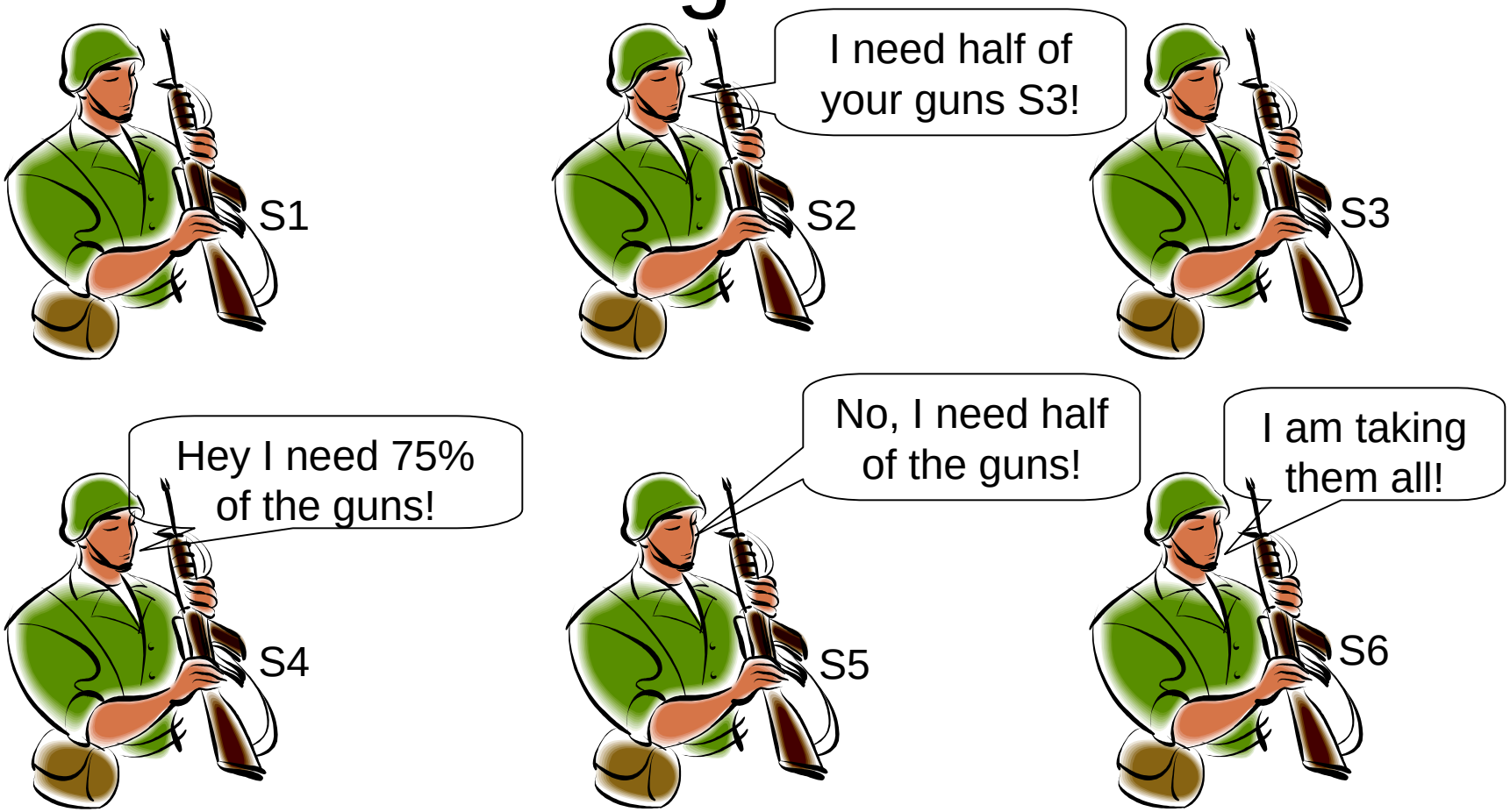
Applications shown are CORBA objects on a host

Centralized Resource Management

- A centralized entity makes resource allocation decisions

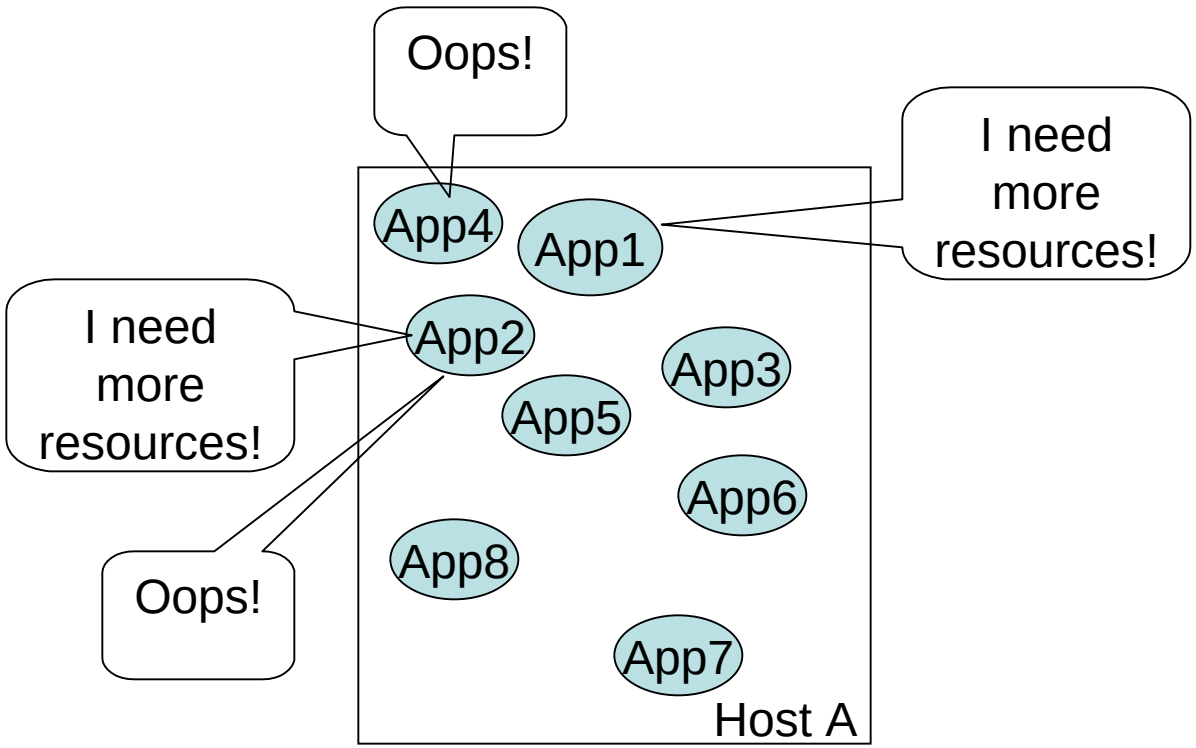
- Issues:
  - Lack of scalability
  - Performance bottleneck

Decentralized Resource Management



S: Soldier

Decentralized Resource Management



Applications shown are CORBA objects on a host

Decentralized Resource Management

- Applications autonomously allocate resources

- Issues:
  - No global perspective
  - May lead to instability
  - Anarchic!

Hybrid Approach

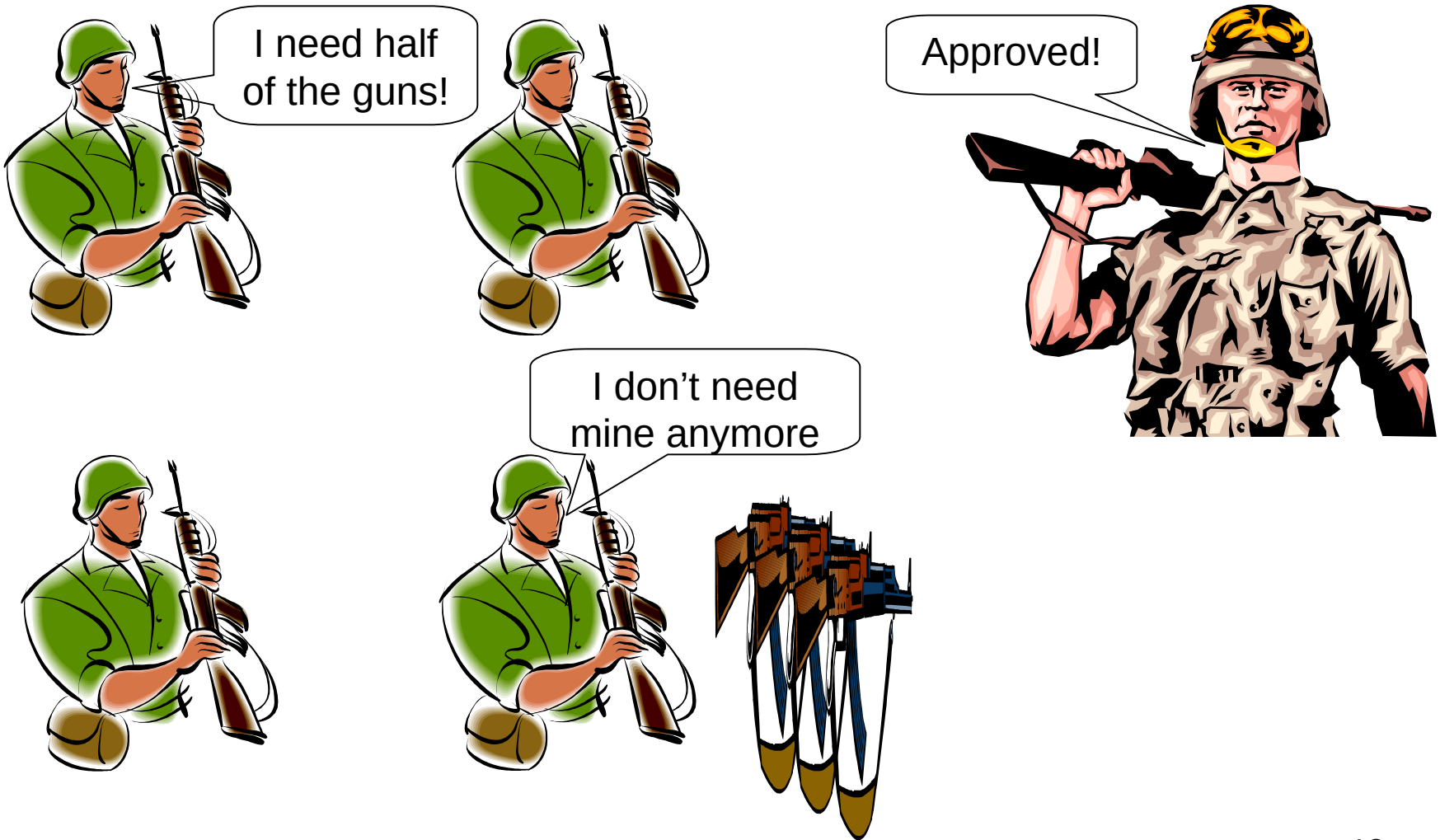
- “Best of both worlds”

- Each real-time application
  - monitors its resource utilization and real-time performance
  - may make requests to
    - Reserve resources
    - Adjust its Quality of Service (QoS) level

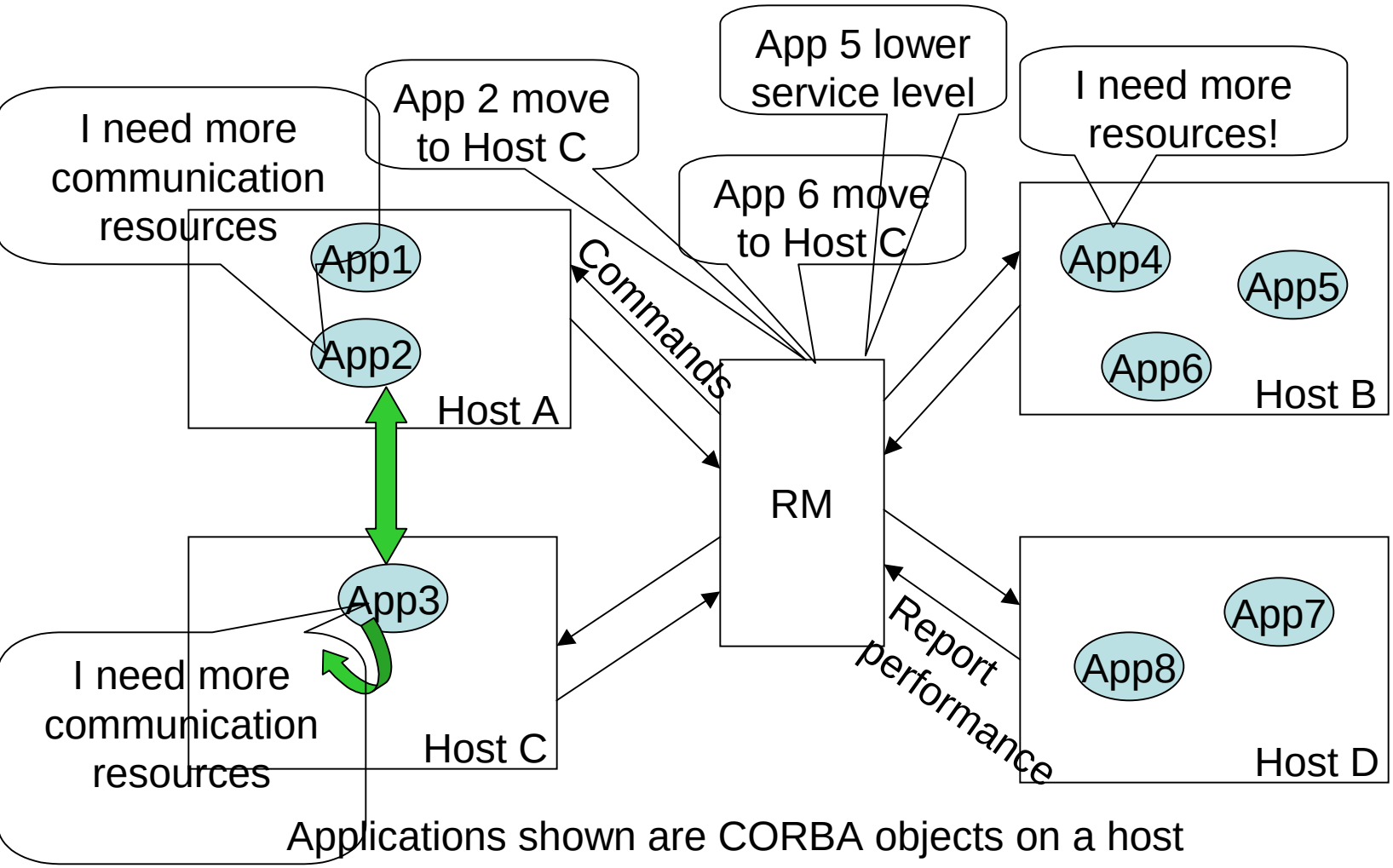
Hybrid Approach

- Resource reservation requests are intercepted by a Resource Manager (RM)
  - RM performs feasibility, optimality and stability analysis
  - RM evaluates if resource reservation request may be granted
  - RM notifies the applications of its decisions

Hybrid Approach

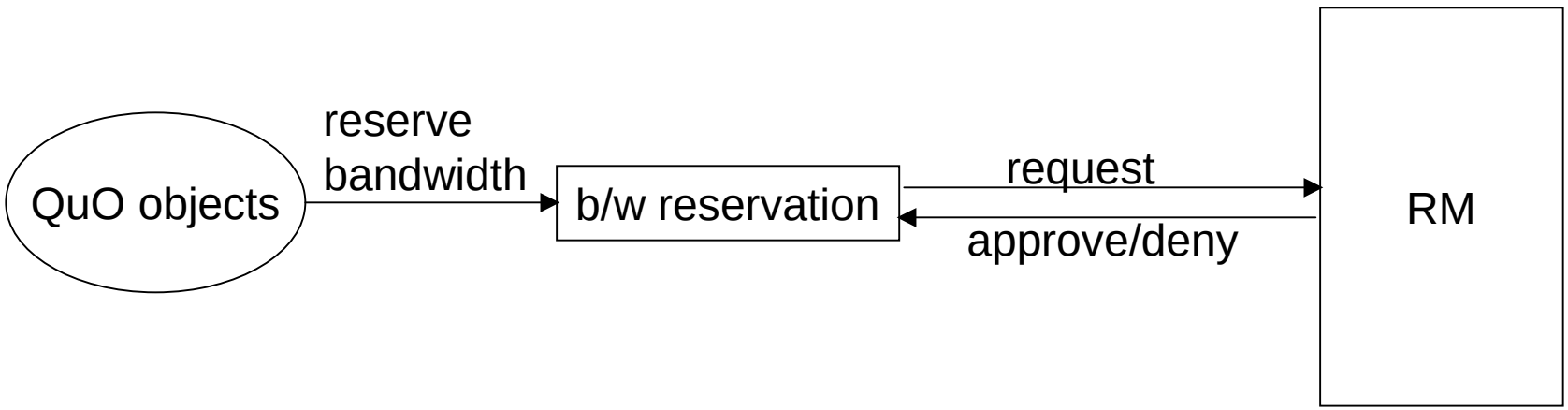


Hybrid Approach



Applications shown are CORBA objects on a host

An Example

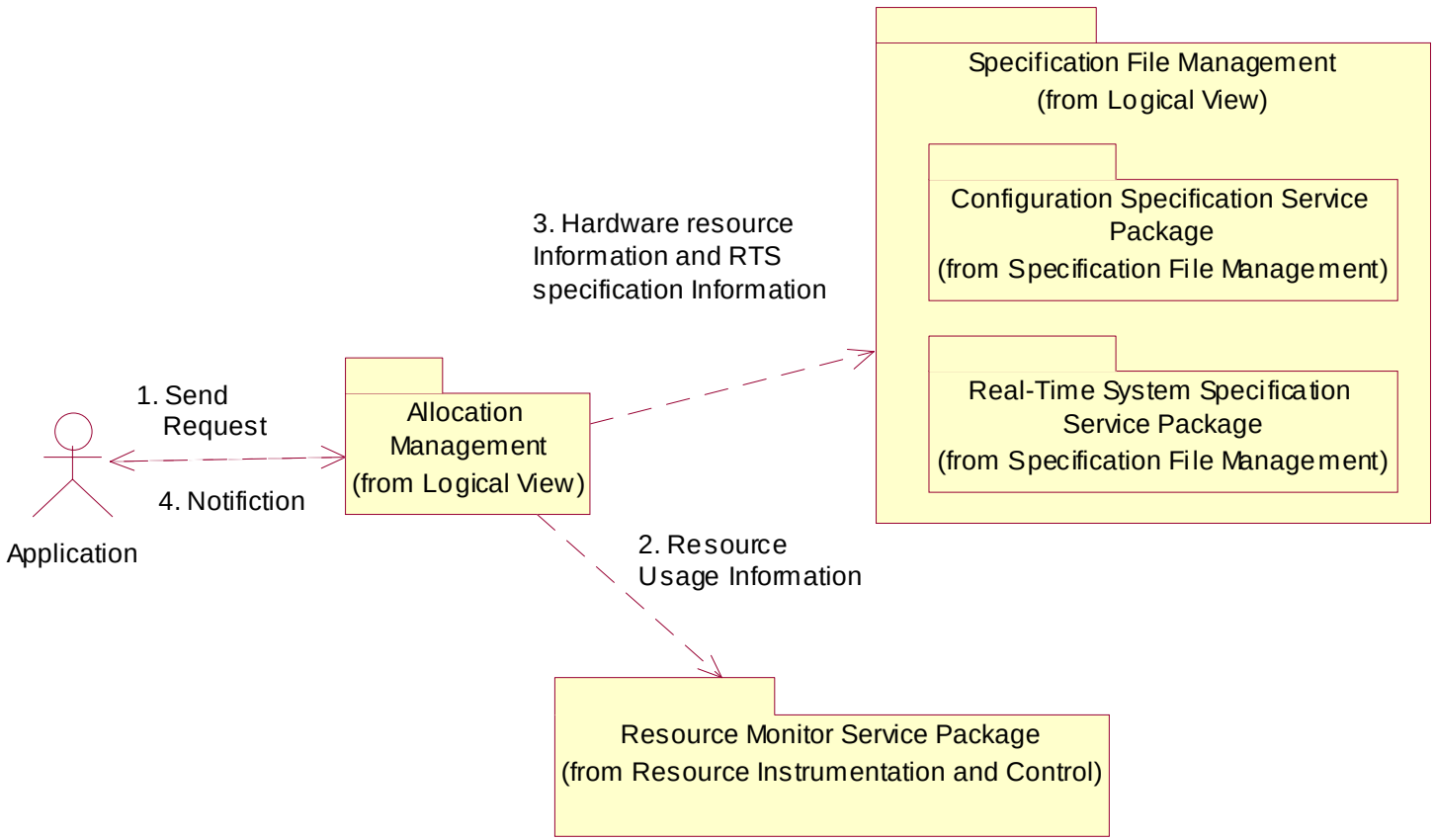

QuO objects
reserve
bandwidth
b/w reservation
request
approve/deny
RM

Handle Resource Reservation Use Case

Application

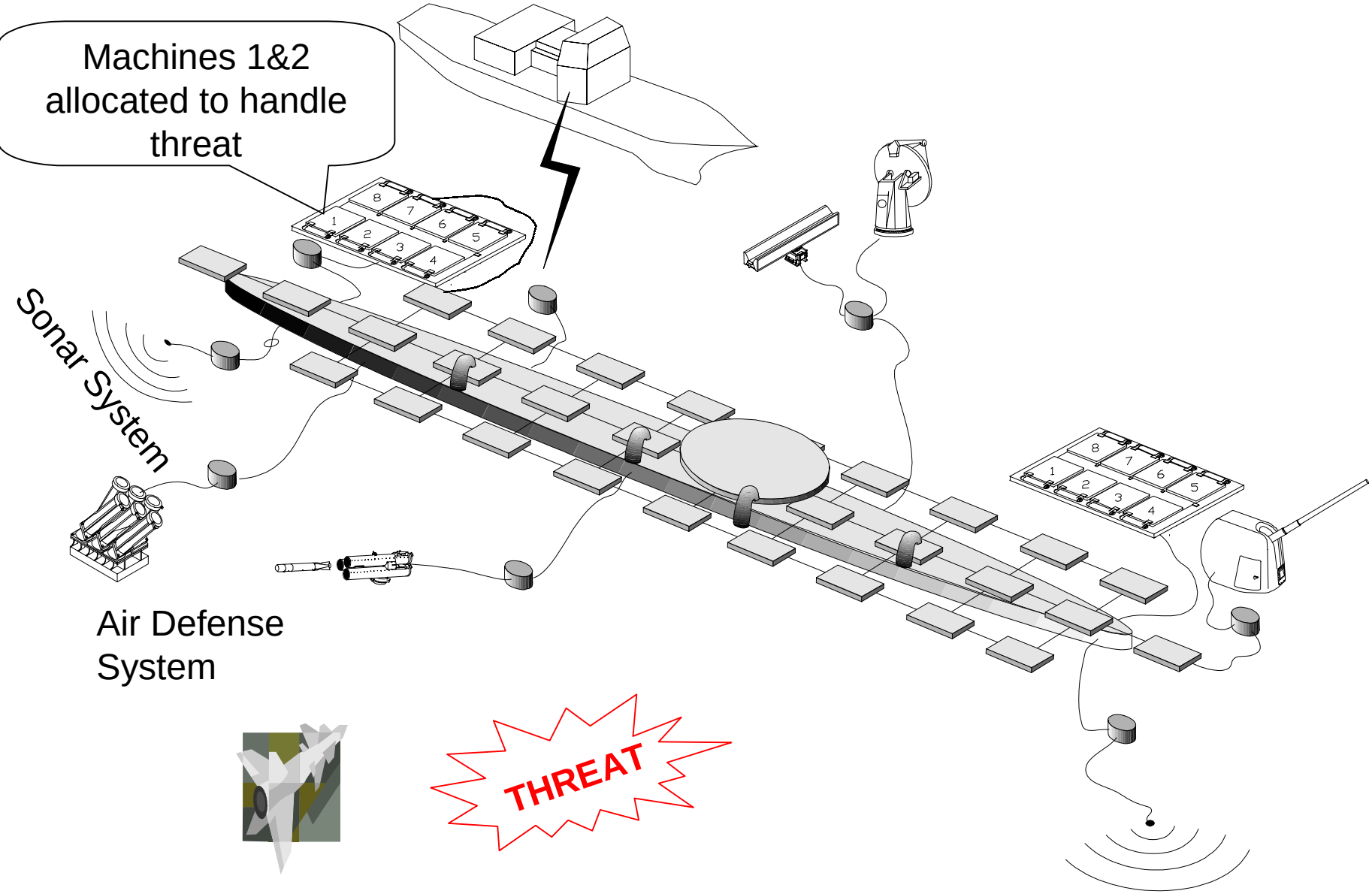
Handle Resource Reservation Request from Applications

Subsystem Collaboration Diagram



Drawbacks

- No notion of relative importance of real-time applications and systems
- User has no high level and dynamic control over resource allocation


Machines 1&2 allocated to handle threat
Sonar System
Air Defense System
THREAT

Need for priorities

- RM needs to be augmented so that:

  A system at a higher priority may achieve higher resource utilization and QoS levels than a system at a lower priority

Advantages of priorities

- User-defined priorities define the relative importance of the real-time systems

- This affords the user a priority-based control *at run-time* over resource allocation among competing systems

RM with Priorities

- User defines priorities
- Applications make resource reservation requests
- RM intercepts these requests
- RM decides to grant/deny the request based on priorities
- RM can revoke resources if priorities change

RM with Priorities

I need half of the guns!

Priority: 4

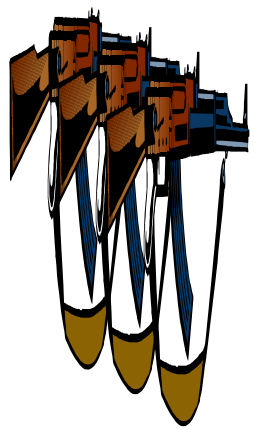
I need half of the guns!

Priority: 3

Priority 1, give back your guns!

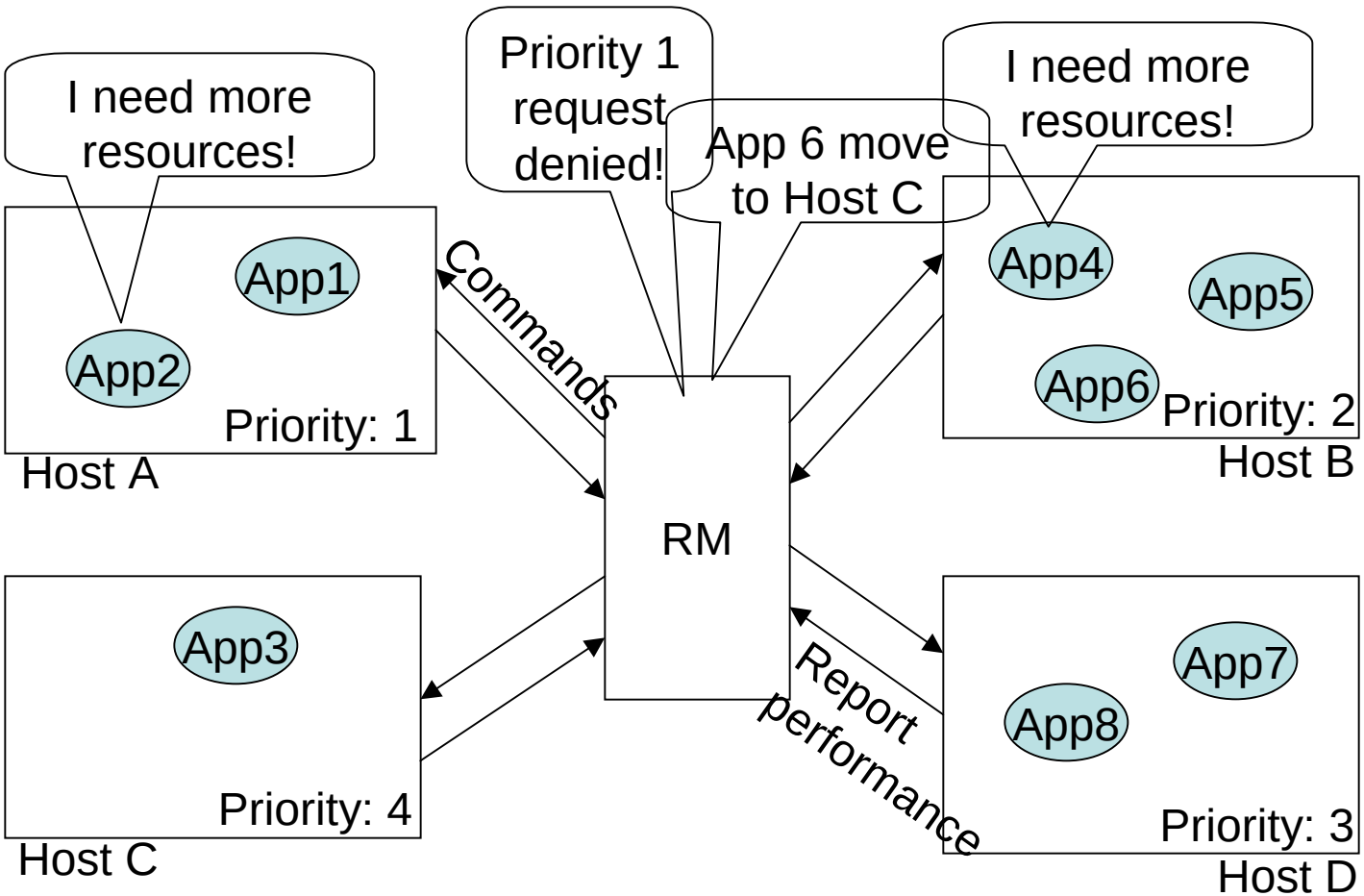
Priority: 2

Priority: 1



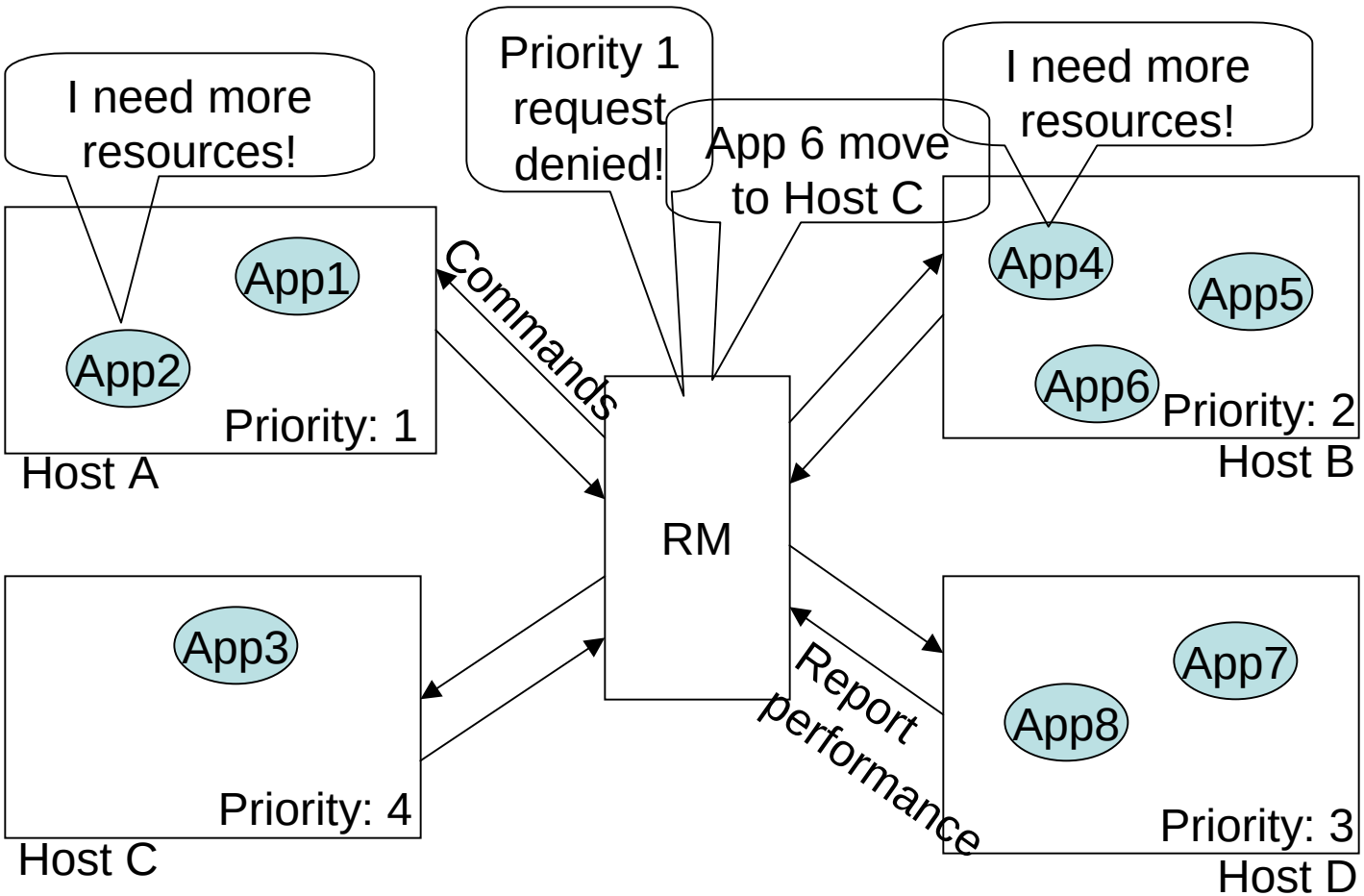
Priority 3, request denied!

RM with Priorities



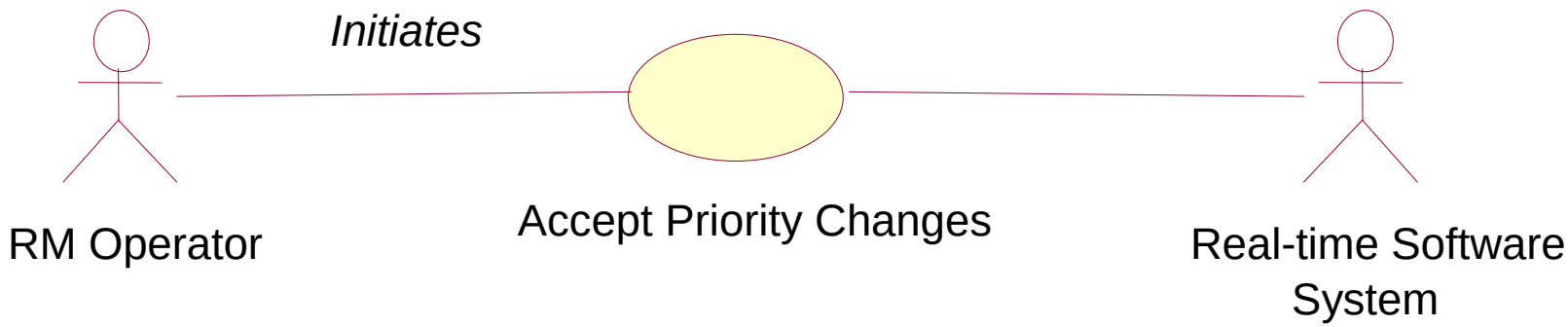
Applications shown are CORBA objects on a host

RM with Priorities

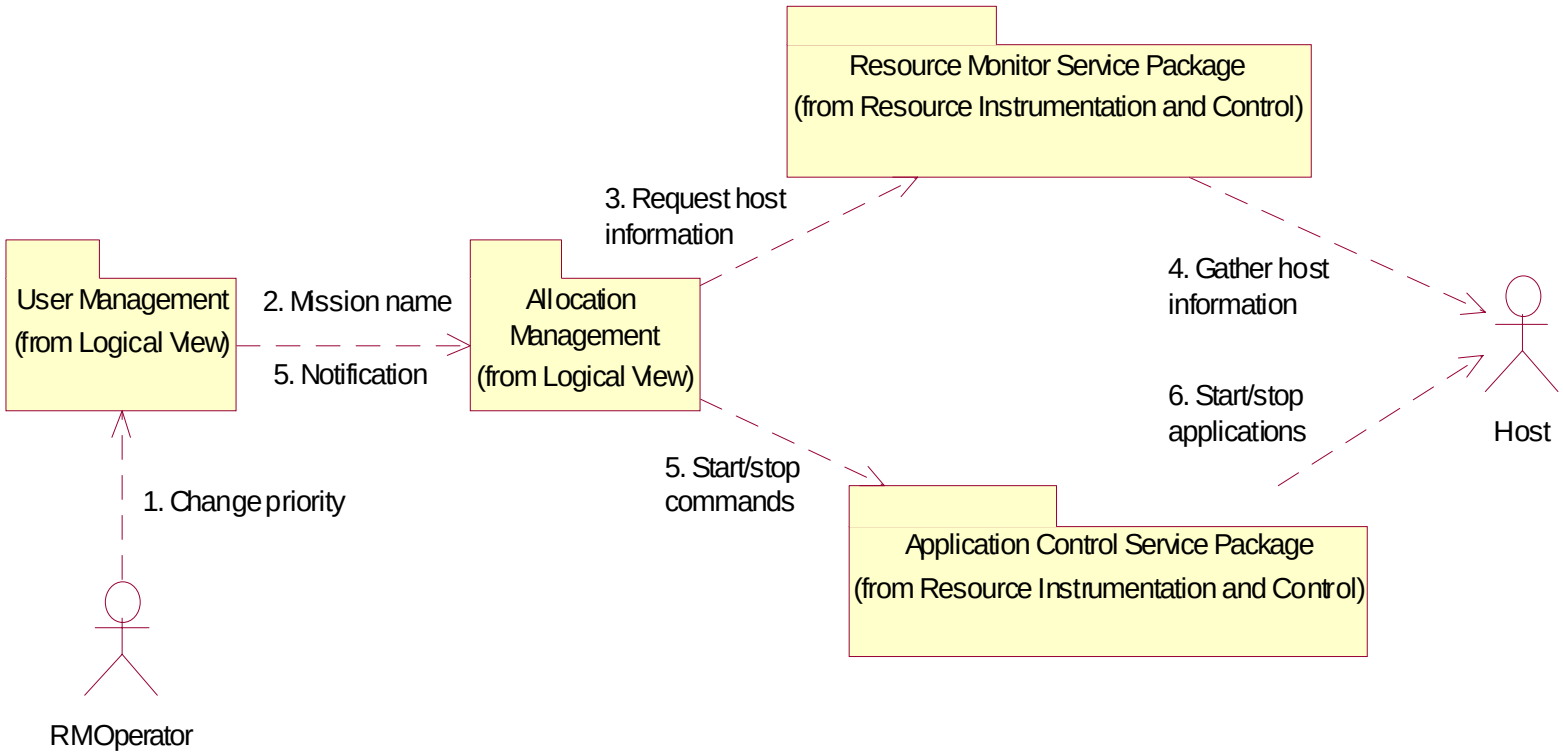


Applications shown are CORBA objects on a host

Accept Priority Changes Use Case



Subsystem Collaboration Diagram



Hybrid Approach: Optimization Cost Function

$$Cost = \max[\sum_{k}(p(A) * utility(A_k))]$$, such that

$$\forall_{r \in R} U(r) < 100\%$$

$$\forall_{A} \lambda_{obs}(A) < \lambda_{req}(A)$$

Where,

A = application

p(A) = priority of the application A

A_k = service level at which application A is running

Utility(A_k) is the benefit accrued by running the application A at service level k.

R is a set of hardware resources

Software Specification File

```
Software System Air Defense {

Priority 10;

Path Sensing {
    Connectivity {
        (sensor, FM)
        (FM, filter)
        (filter, EDM)
        (EDM, ED)
        }
    RealTimeQoS {
        Deadline 0.3;
    }

}
```

Software Specification File

```
Path Guidance {
    Connectivity {
                    (MGM, MG)
        }
        RealTimeQoS {
                    Deadline 0.2;
            }
    }

Path  Action{
    Connectivity {
                (AM, action)
                (action, actuator)
        }
        RealTimeQoS {
                Deadline 0.2;
        }
    }
```

Software Specification File

```
Software System Sonar{
Priority 5;
Application surveillance {
    Period = 1 sec;
    Service Level 1{
        utility = 1.0;
        CPU = 0.25 sec;
    }
    Service Level 2{
        utility = 0.5;
        CPU = 0.15 sec;
    }
    Service Level 3{
        utility 0.2;
        CPU = 0.10 sec;
    }
}
```

Conclusions

- Combines the advantages of centralized and decentralized resource management approaches
- Allows for user-defined priorities
- Hybrid approach provides an effective way of managing the resources as well as responding to the dynamic environment