A Design Methodology for Self-Management in Distributed Environments

Ahmad Al-Shishtawy[1] Vladimir Vlassov[1] Per Brand[2]
Seif Haridi[1,2]

1 - Royal Institute of Technology (KTH), Stockholm, Sweden

2 - Swedish Institute of Computer Science (SICS), Stockholm, Sweden

ahmadas@kth.se

SEC09, Vancouver, Canada
August 31 2009

Outline

Outline

Autonomic Computing

- Complex applications cause high administration overheads
- Autonomic computing is a paradigm that aims at:
  - reducing administration overhead
  - enabling new complex applications

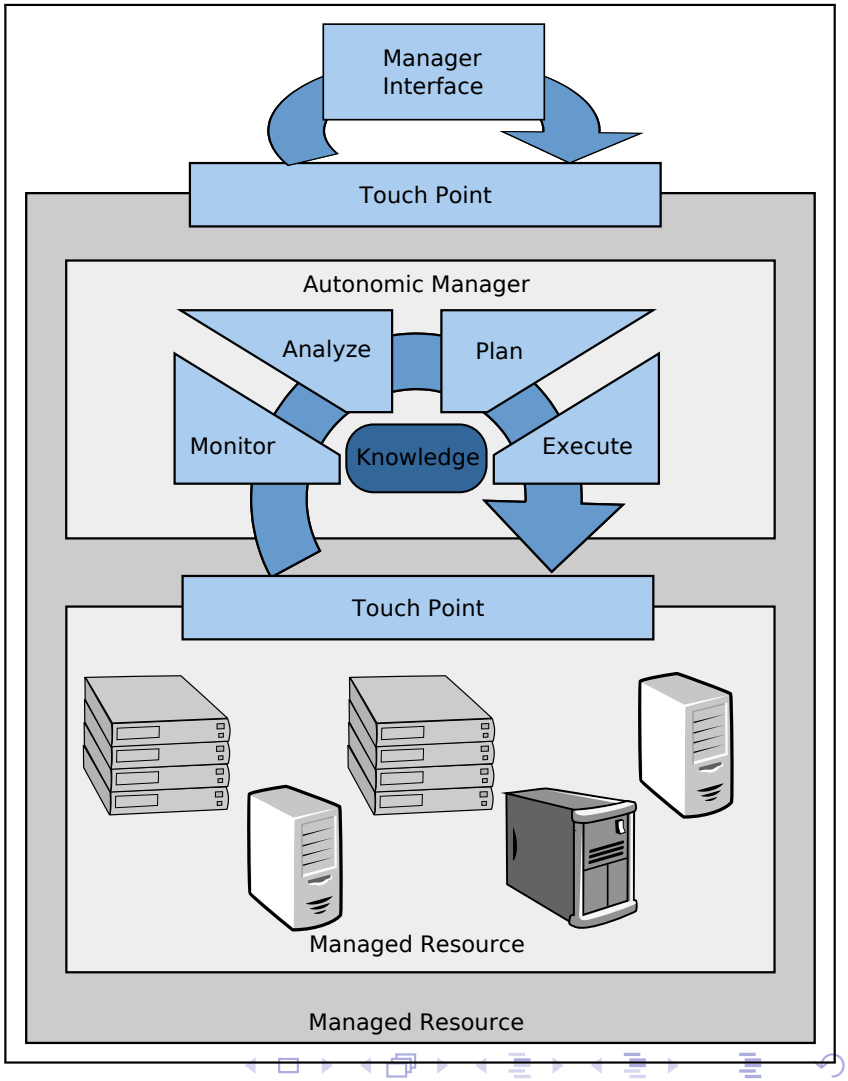
  by introducing self-management, i.e. building systems and applications that are able to manage themselves

Self-* Properties

- Inspired by the autonomic nervous system of the human body
- Control loops from Control Theory
- Self-management includes four self-* properties:
  - self-configuration
  - self-optimization
  - self-healing
  - self-protection

The Autonomic Computing Archecture

- Managed Resource
- Touchpoint
  - Sensors
  - Actuators
- Autonomic Manager
  - Monitor
  - Analize
  - Plan
  - Execute
- Knowledge Source
- Communication
- Manager Interface



Distributed Management

- In distributed environments we advocate for distribution of management functions among several cooperative managers
- Multiple managers are needed for scalability, robustness, and performance and also useful for reflecting separation of concerns
- Need guidance on how to design distributed management

Outline

What is Niche

Niche is Distributed Component Management System (DCMS)

- URL: `http://niche.sics.se/`
- Implements the autonomic computing architecture
- Includes a component-based programming model (Fractal) , API, and an execution system
- Allows development, deployment, and execution of self-managing distributed component-based applications
- Can be used in dynamic distributed environments
- Seperates programing of functional and management parts
- Provides transparant replication of management for robustness

Niche Implementation

- Uses a structured overlay network (SON) for communication.
- Provides sensors and actuators.
- The Autonomic Manager in Niche is organized as a network of Management Elements (MEs)
  - Watchers
  - Aggregators
  - Managers
  - Executors
- Knowledge is shared between MEs using pub/sub or DHT provided by the underlying SON.

Outline

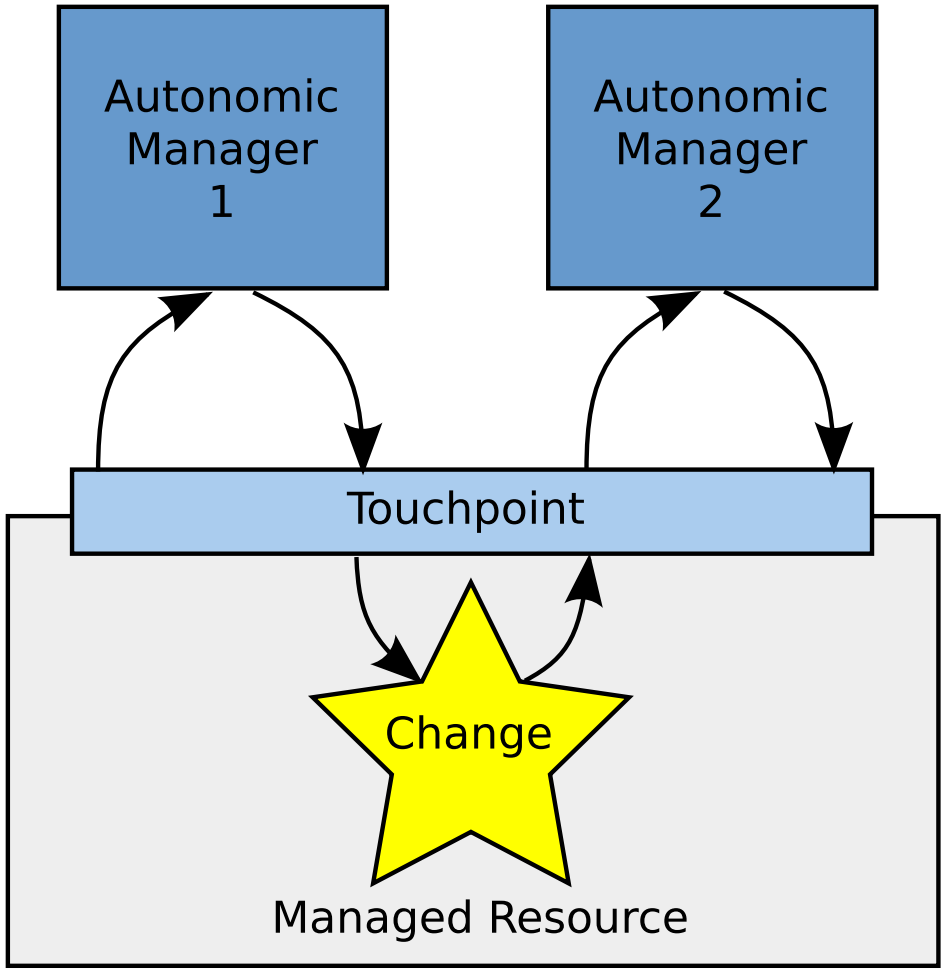
High Level Design Steps

- A self-managing application consists of three parts:
  - Functional part
  - Touchpoints
  - Management part
- Iterative steps to distribute management (given management objectives):
  - Decomposition
  - Assignment
  - Orchestration
  - Mapping

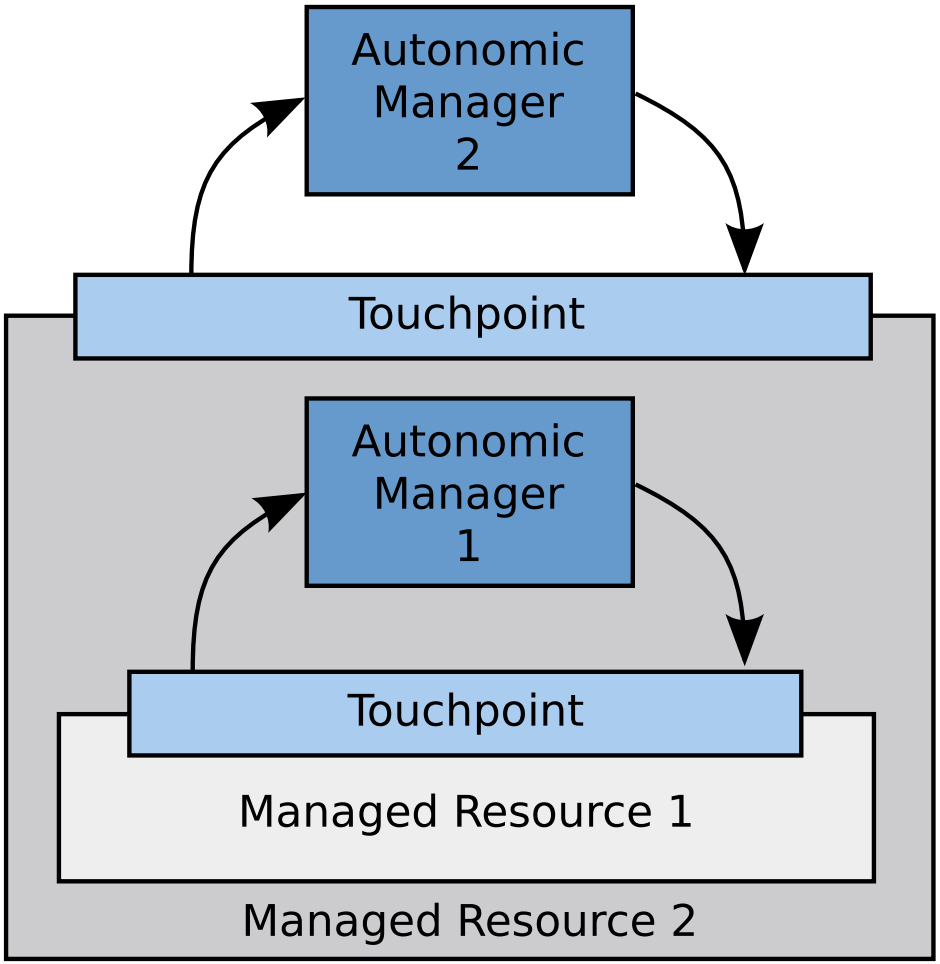
Design Space for Management Interaction

- Stigmergy
- Hierarchical
- Direct Interaction
- Sharing of MEs

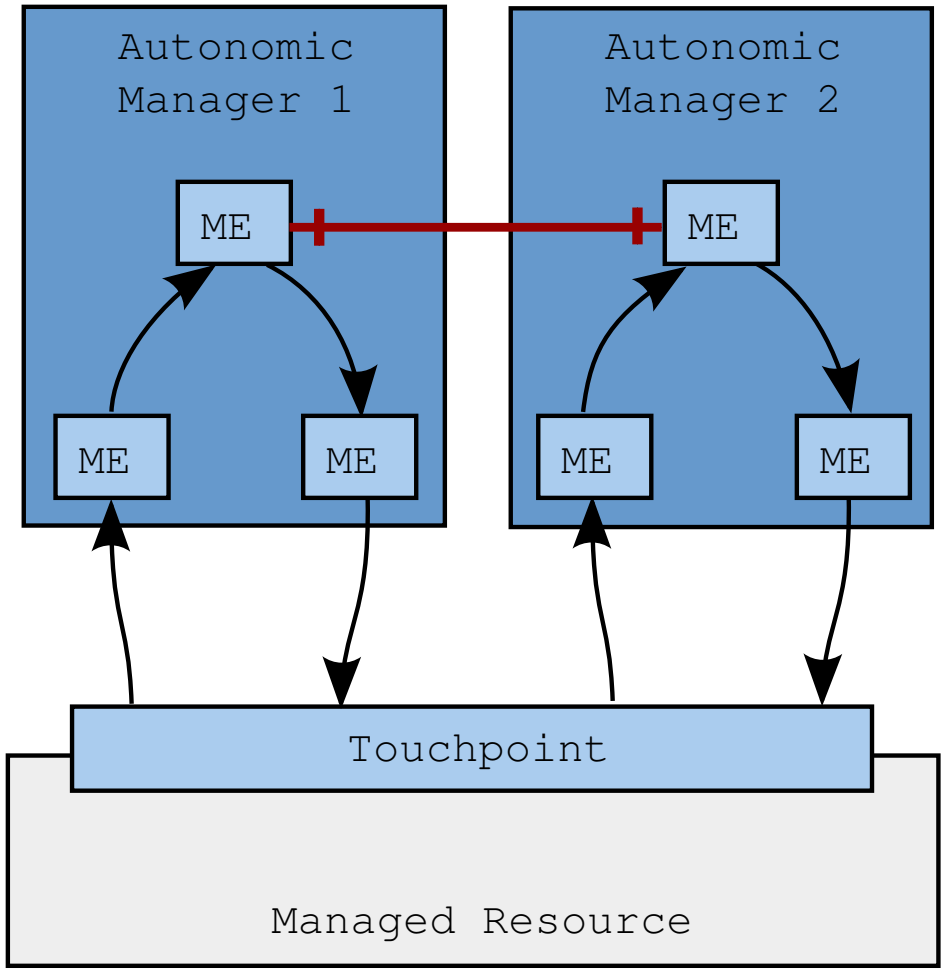
Stigmergy


Autonomic Manager 1
Autonomic Manager 2
Touchpoint
Change
Managed Resource

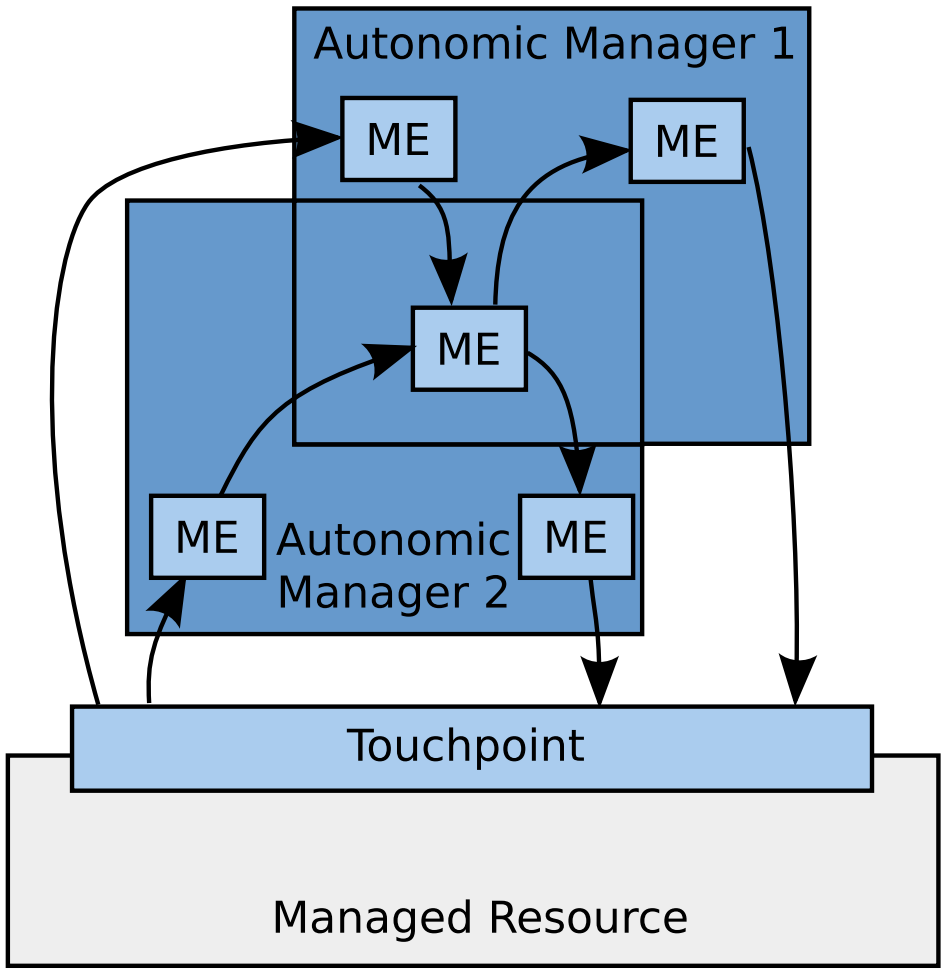
Hierarchical



Direct Interaction



Sharing of MEs

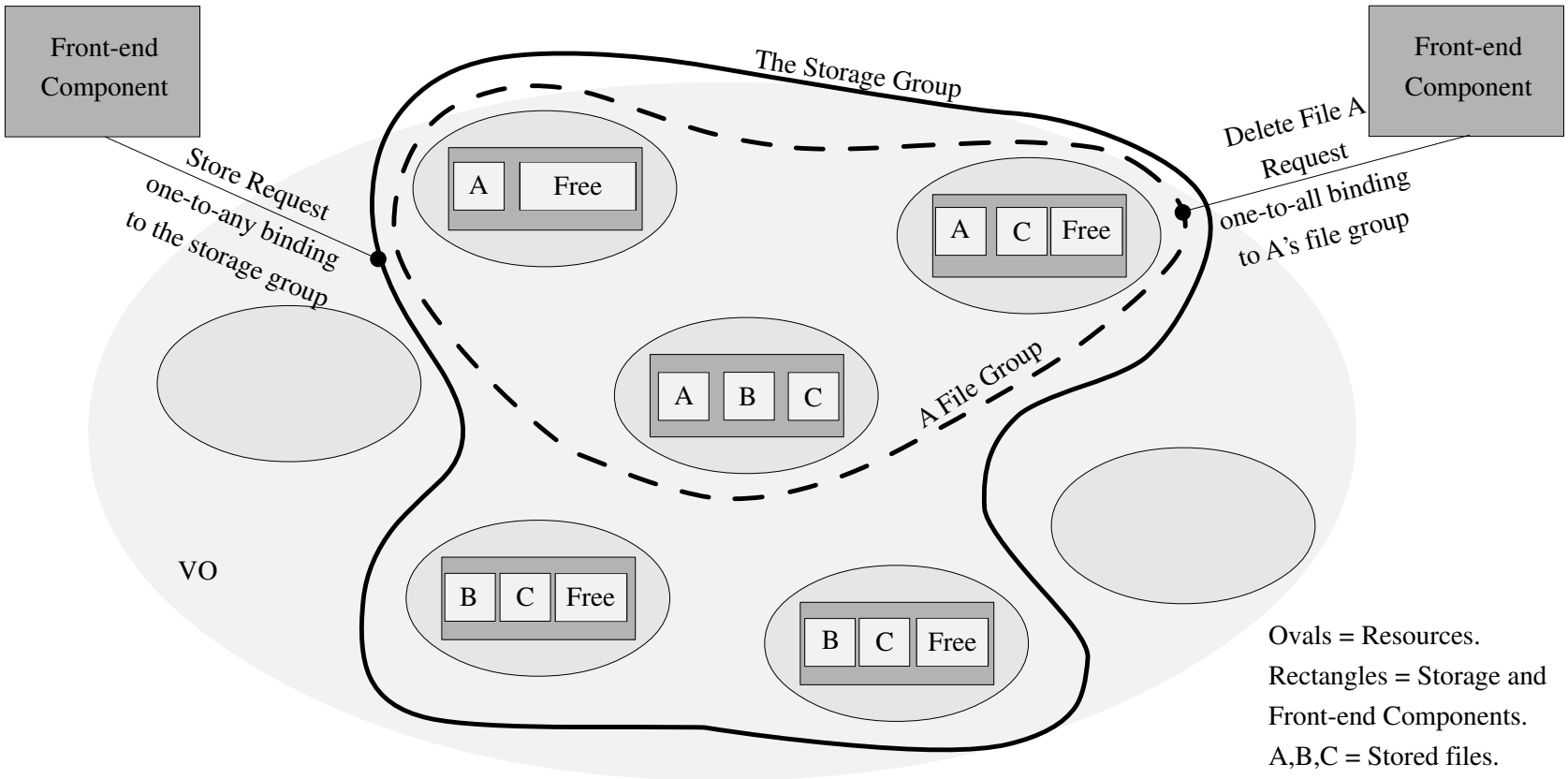


Outline

What is YASS

- YASS: Yet Another Storage Service
- Users can store, read and delete files on a set of distributed resources.
- Transparently replicates files for robustness and scalability.
- Deployed in a dynamic distributed environment

YASS functional part



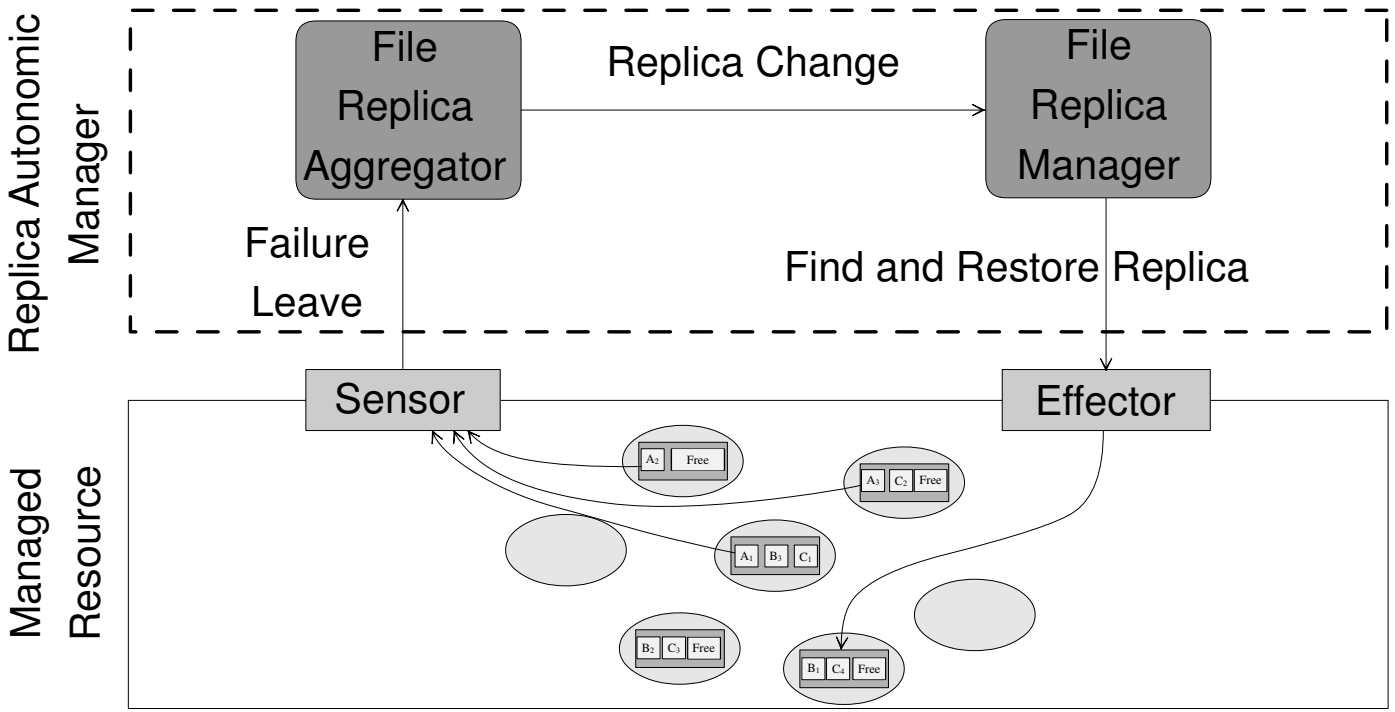
Management Requirements

- Maintain the file replication degree
- Maintain the total storage space and total free
- Increasing the availability of popular files
- Release extra allocated storage
- Balance the stored files among the allocated resources

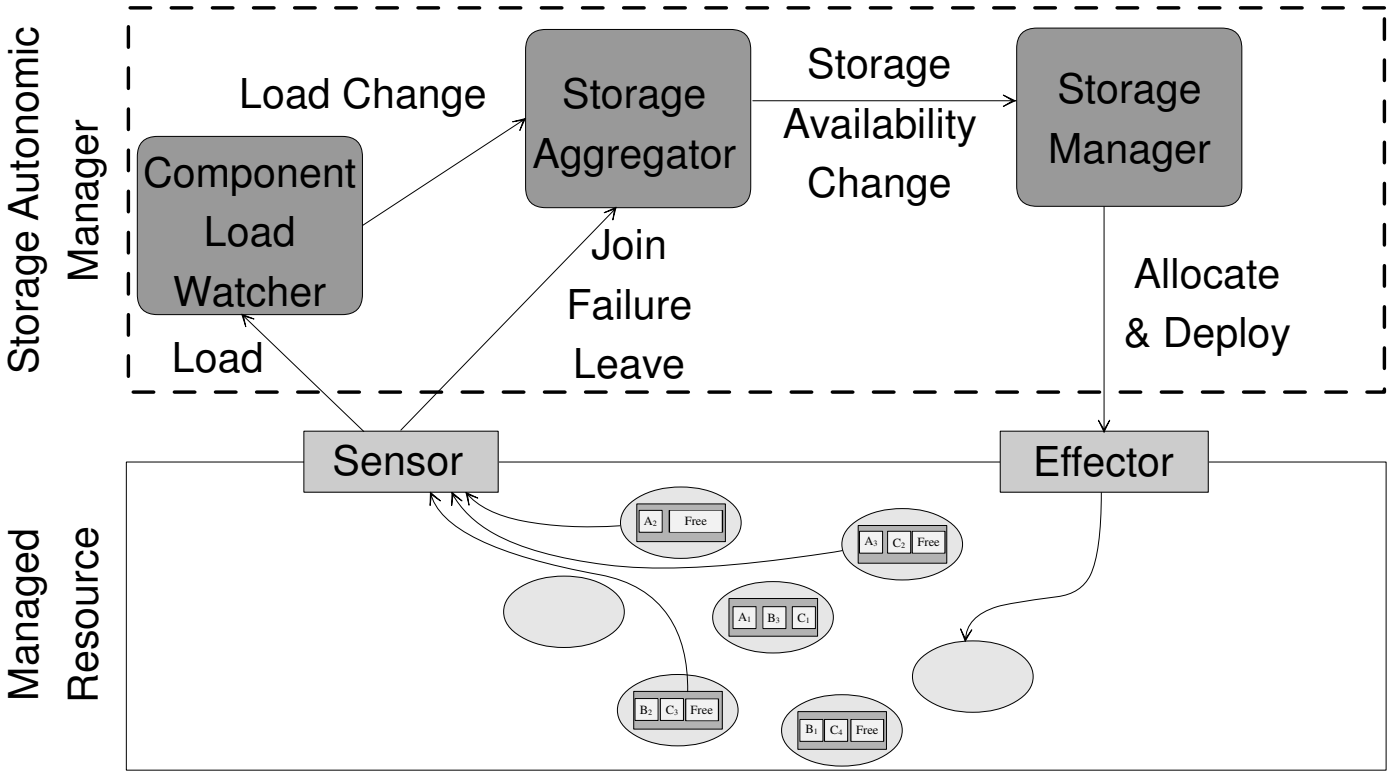
Touchpoints

- Load sensor to measure the current free space
- Access frequency sensor to detect popular files
- Replicate file effector to add one extra replica of a file
- Move file effector to move files for load balancing

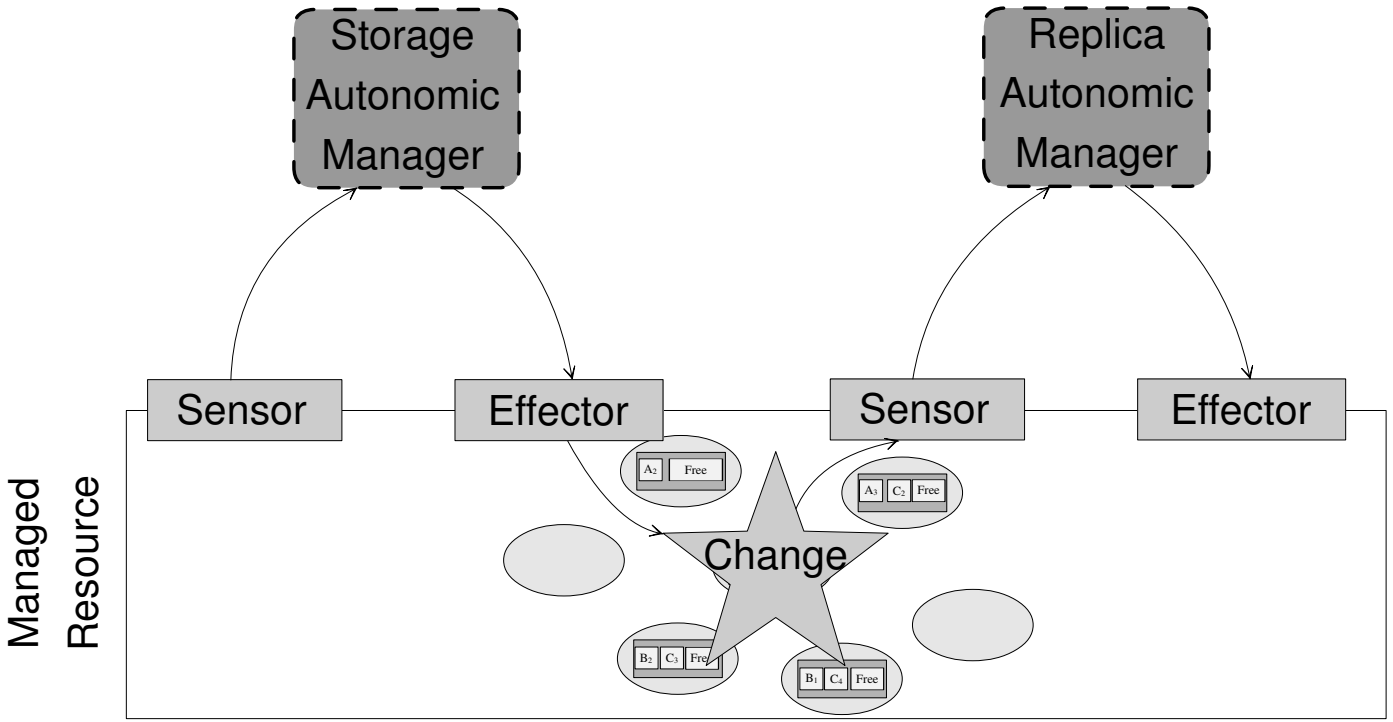
Maintain the file replication degree



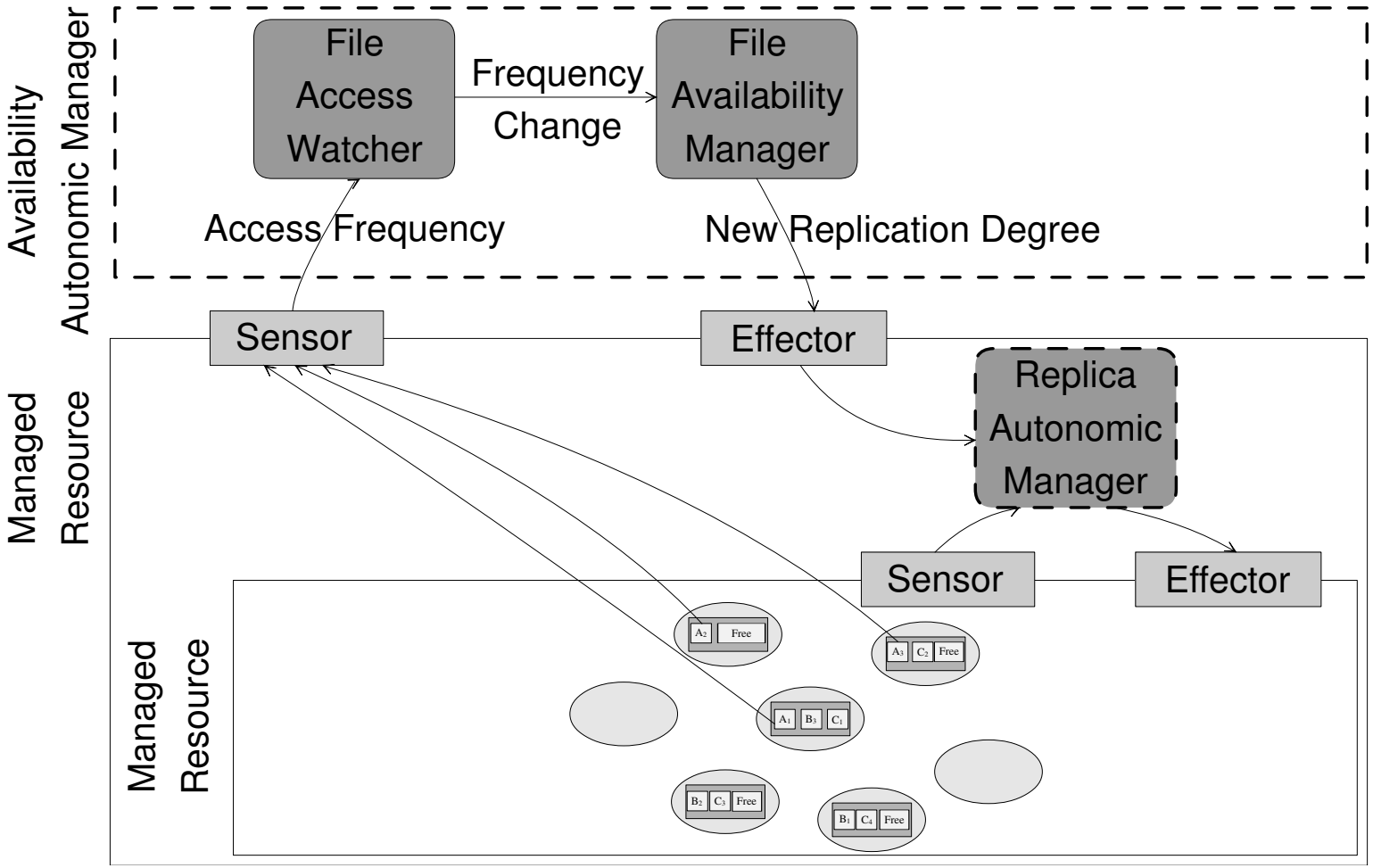
Maintain the total storage space and total free



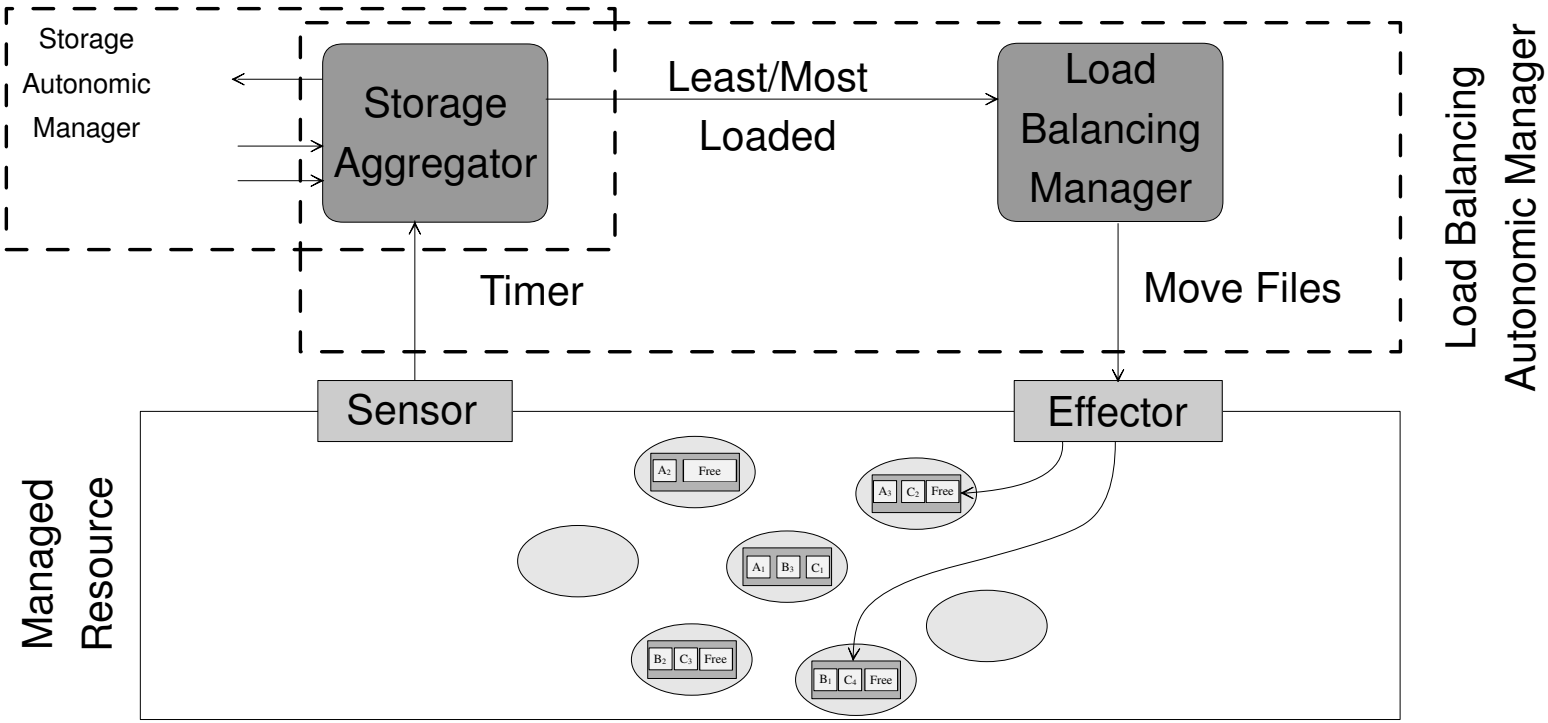
Release extra allocated storage



Increasing the availability of popular files



Balance the stored files among the allocated resources



Outline

Conclusions

- Distribution of management will improve scalability and performance
- Presented the steps for developing distributed management
- Defined design space of manager interactions that includes four patterns used to orchestrate autonomic managers
- Applied the design methodology to a distributed file storage application

Future Work

- Refine our design methodology and identify reusable patterns
- Performance evaluation through simulation and experiments on PlanetLab
- Language support (e.g. policy languages, coordination language, and ADL)
- Consider more use cases

Thank you for careful listening :-)

Questions?