Introduction to adaptive computing systems

Vania Marangozova-Martin
Associate Professor, University of Grenoble – LIG, France

Vania.Marangozova-Martin@imag.fr

Web Site: http://acs.forge.imag.fr

Administrivia

- 10 weeks, 1 lecture, 1 lab per week
- Lab demonstrations
- Article presentations
- Final exam

Outline of this lecture

- Some taxonomy
- Computing systems
  - and the need for adaptation
- Notion of service and interface
- Adaptation: what and how
  - interceptors
  - MOP

Adaptation of computer systems : a simple taxonomy

- Three main questions:
  - What for?
    - Why would we need to adapt the system?
      - Software engineering : better quality of code for maintability, reuse, evolutivity…
      - Examples
      1) The entreprise changes its scale and passes from 100 to 10000 workers, production is +1000%, some parts are to be delocalized…
      2) Web application with important visual aspect and the integration of new, more performant technologies

Adaptation of computer systems : a simple taxonomy

- Three main questions:
  - What for?
    - Why would we need to adapt the system?
      - Software engineering : better quality of code for maintability, reuse, evolutivity…
      - Better functionality : performance evaluation
  - What is adapted?
  - How is it adapted?

Adaptation of computer systems : a simple taxonomy

- Three main questions:
  - What for?
    - Why would we need to adapt the system?
      - Software engineering : better quality of code for maintability, reuse, evolutivity…
      - Performance evaluation : performance optimization, scaling
      - Examples

1) System becomes popular and needs to manage 1,000,000 users (instead of 100…)
2) System migrates from 10-machine cluster to a 1000-machine cluster

Adaptation of computer systems : a simple taxonomy

- Three main questions:
  - What for?
    - Why would we need to adapt the system?
      - Software engineering : better quality of code for maintability, reuse, evolutivity…
      - Performance evaluation : performance optimization, scaling
      - Context change : new requirements for system execution
      - Examples

1) System needs to support mobile users
2) System needs to integaret security and confidentiality features

Adaptation of computer systems : a simple taxonomy

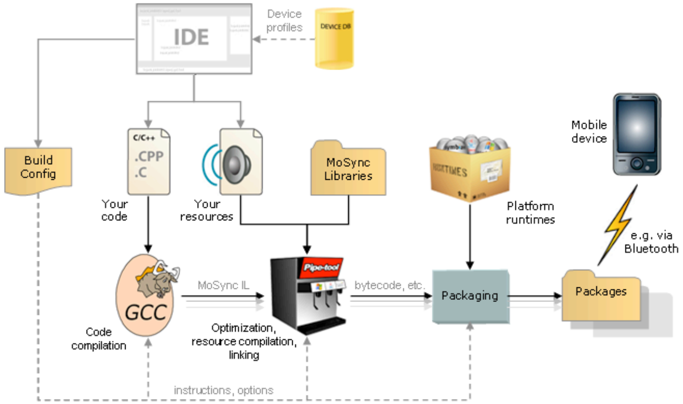
- Three main questions:
  - What for?
    - The requirements for a system are constantly evolving
  - How?
- change the code
  - change the code to change its functionality

  - redevelop the code with suitable optimizations
    - compilation chain

Diversity and Adaptation

- How to adapt the application:
  - redevelop the code with suitable optimizations
  - compilation chain

Compilation Chain Example



Source: http://www.mosync.com/docs/sdk/tools/guides/architecture/toolchain/index.html

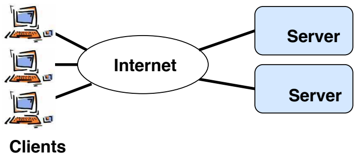
Adaptation of computer systems : a simple taxonomy

- Three main questions:
  - What for?
    - The requirements for a system are constantly evolving
  - How?

- change the code
- change the architecture
  - higher level of abstraction
  - coarse grain
  - semantics, interfaces

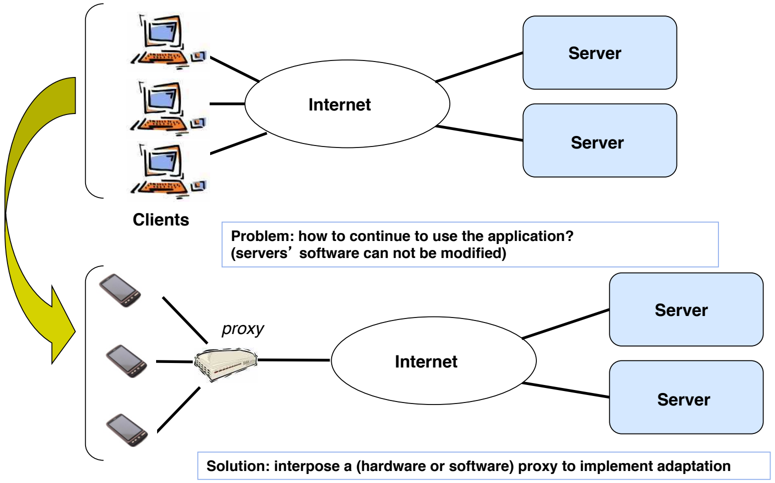
Adaptation of computer systems : a simple taxonomy

- Three main questions:
  - What for?
    - The requirements for a system are constantly evolving
  - How?



- change the architecture
  - higher level of abstraction
  - coarse grain
  - semantics, interfaces

Example : architecture adaptation with interceptors



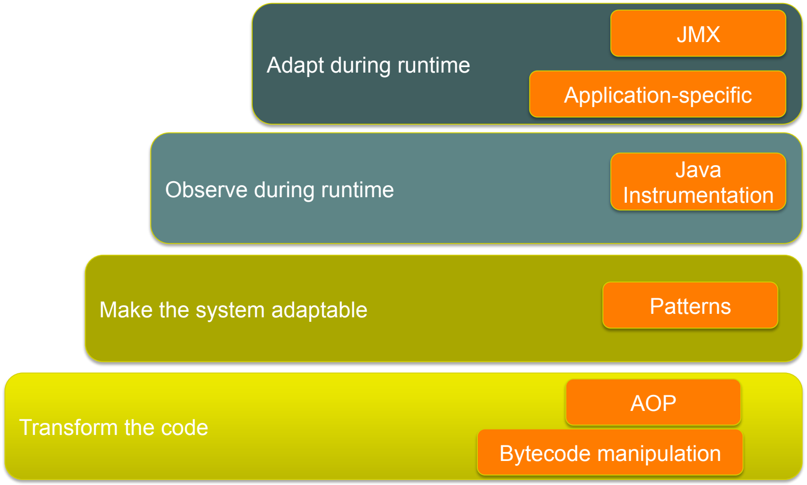
Problem: how to continue to use the application?
(servers' software can not be modified)

Solution: interpose a (hardware or software) proxy to implement adaptation

Adaptation of computer systems : a simple taxonomy

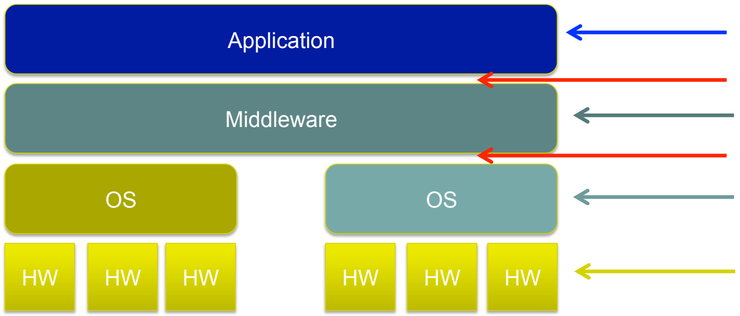
- Three main questions:
  - What for?
    - The requirements for a system are constantly evolving
  - How?
    - change the code or the architecture
  - When?
    - statically: the system does not execute, teh adaptation is done, and the software deployed
    - dynamically: during runtime
  - What mechanism?
    - A large diversity
    - Adaptation is conditioned/defined/made possible by what is adaptable

In this course

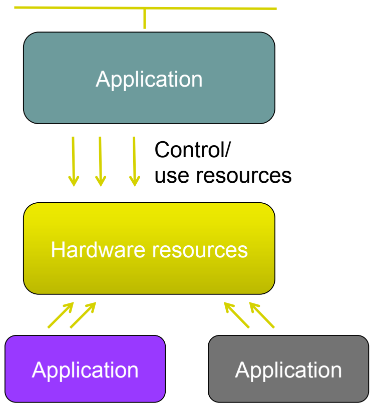


Another view at adaptation…

- At which level to adapt?

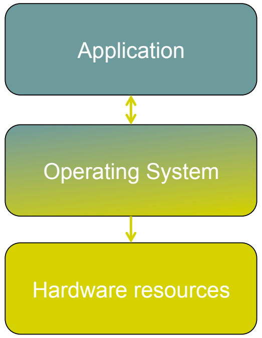


A Computer System



- An application
  - role: answer to a specific problem
  - provides a function, services, to its end-users (or other applications)
- Executes in a given platform
  - The application strives for optimal exploitation of the available resources
  - The platform needs to be exploited optimally

Computer System Layers



- Mapping of application components to physical resources is typically controlled by static OS policies
  - (sometimes) these policies can dynamically change the mapping
    - E.g virtual page replacement policies
  - (rarely) these policies can adjust themselves dynamically
    - E.g switch to one page-replacement policy to another
  - (very rarely) the application can control some policy parameters

Inadequacy of Static Mapping

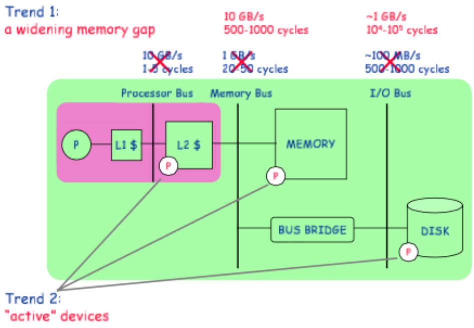
- Collision of two trends
  - Increasing diversity in execution platforms
    - uniprocessor, mutlicore, embedded
    - parallel/distributed
      - CPU capabilities, memory capacities, network characteristics
  - Applications span multiple execution platforms
    - A component needs to interact with other components that run on diverse platforms
    - A component must itself run on mutliple platforms
- Consequence
  - Larger penalties for bad mapping decisions

$$$!!!

Diversity: Uniprocessors

- Widening Memory Gap
  - Cache hierarchies
- Processors become multicore
- Additional speialized processors for memory and disk management
- SSD, HD…
- Energy-efficient processors

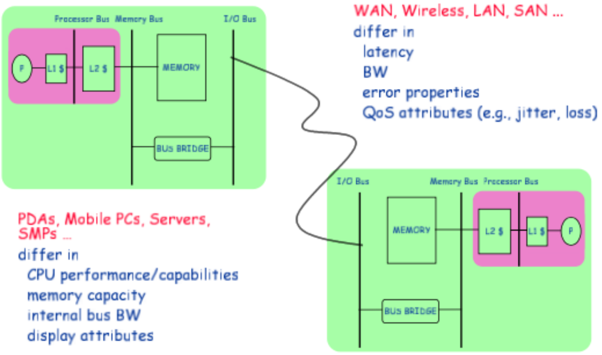
A vision from 10 years ago



- Now
  - Speed is at least x10, P -> multicore, L3 cache

Diversity: Parallel/Distributed Systems

- At the hardware level



Actually, the real picture is more like…

- Kalray MPPA
  - HPC embedded
  - 256 cores
  - Network on Chip (NoC)

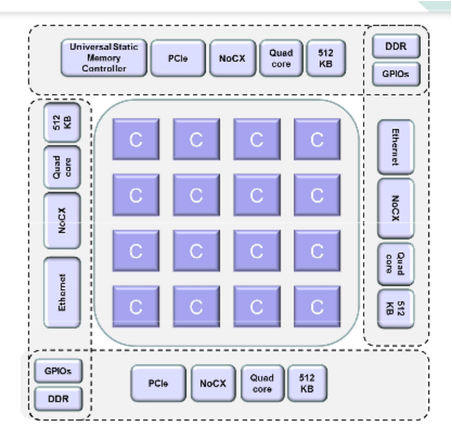
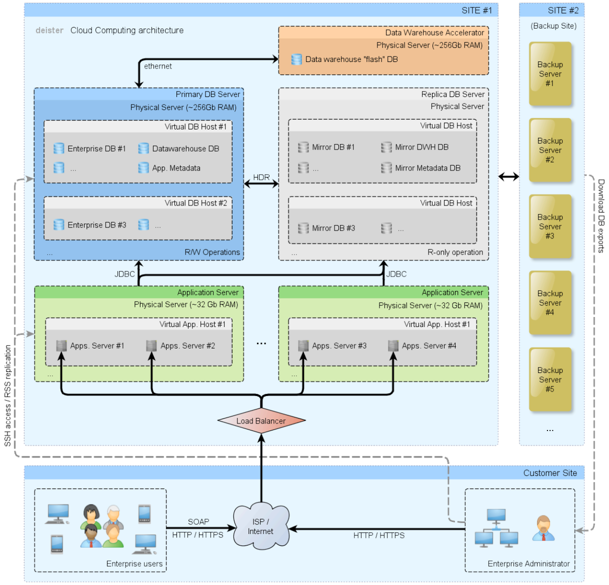
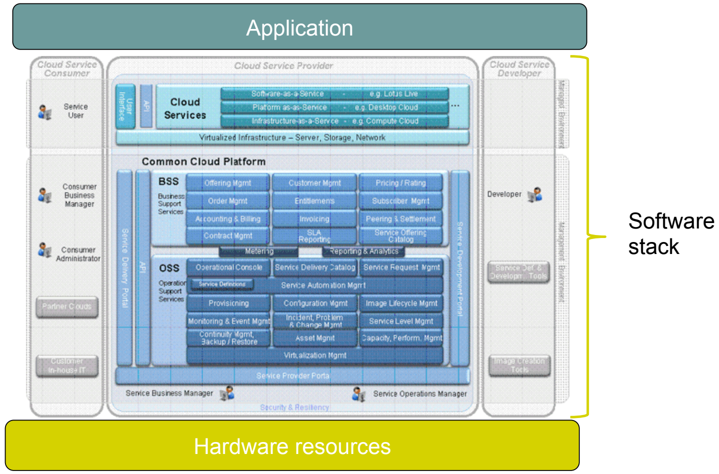


Figure 1 – MPPA®-256 block diagram

Or…



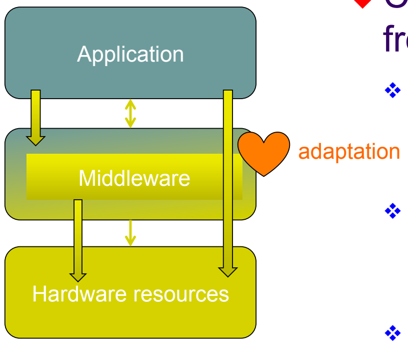
Or…



Adaptation Requirements

- Uniprocessors
  - Same application must execute in diverse environments
  - Binary must
    - Take advantage of available hardware
    - Configure hardware as approprate for the application
    - Tradeoff different resources (eg memory access vs computation)
- Parallel/Distributed systems
  - Application components must interoperate across an even larger range of computing, networking and storage capabilities
  - components may need need to migrate across multiple platforms
  - A harder problem because environment is subject to change at runtime

Computer System Layers: Modern View

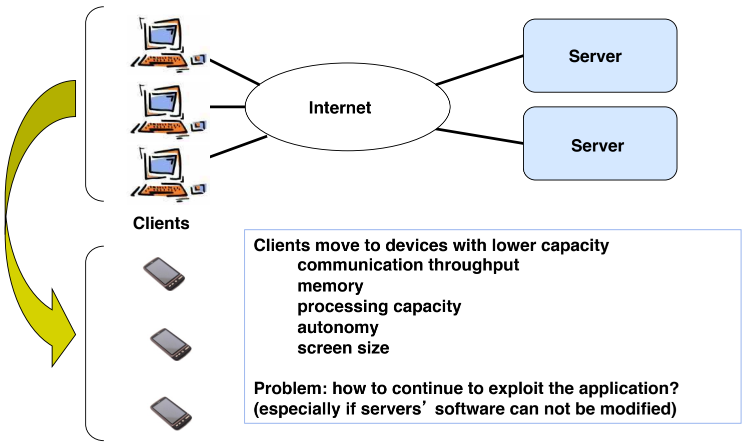


- Controlled depature from layered model
  - Application uses resources based on its requirements
  - Application can adapt to changes in resource availability
  - Improved performance, reliability, use…

Real-world examples for adaptation

- The following examples are so commun and we are so used to having them that we (users) do not really pay attention to/ are not aware of the adaptation issues and the technical complexity

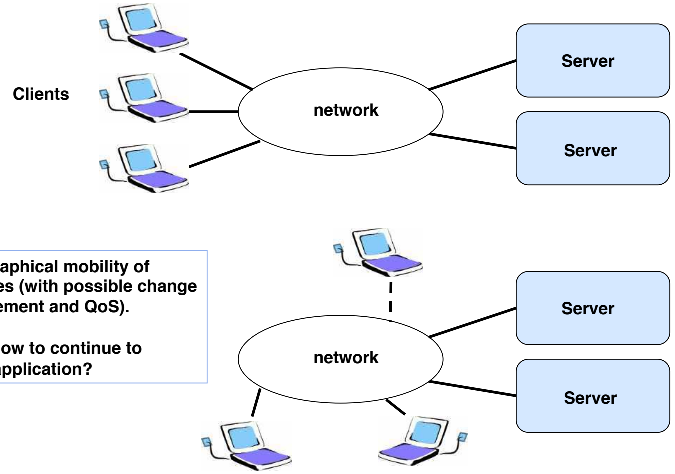
Example 1: Service adaptation based on client device capacity



Clients move to devices with lower capacity
- communication throughput
- memory
- processing capacity
- autonomy
- screen size

Problem: how to continue to exploit the application? (especially if servers' software can not be modified)

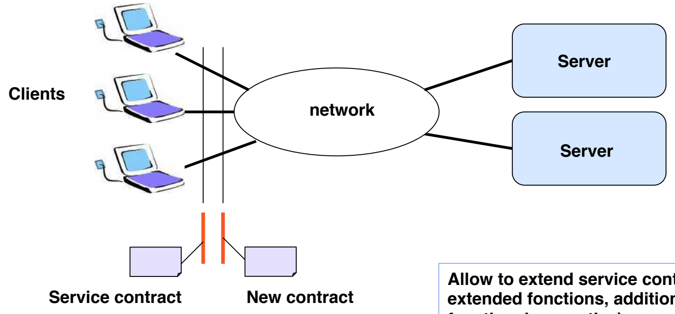
Example 2 : Service adaptation in case of client mobility



Allow geographical mobility of client devices (with possible change in environnement and QoS).

Probleme: how to continue to exploit the application?

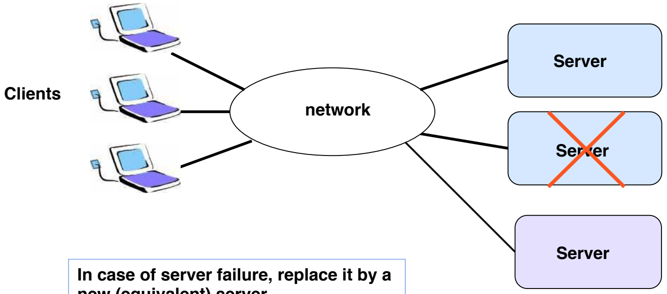
Example 3 : Service extension and evolution



Allow to extend service contract (e.g. extended fonctions, additional non-functional properties)

Problem: how to allow this evolution (without service interruption)?

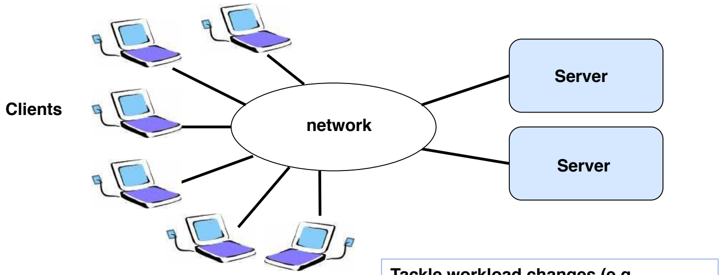
Example 4 : Service adaptation for fault-tolerance



In case of server failure, replace it by a new (equivalent) server

Problem: how to tolerate failures (failure detection, server replacement), without service interruption?

Example 5 :
Service adaptation for workload changes



Tackle workload changes (e.g. #concurrent clients)

Problem: how to maintain an acceptable level of QoS (e.g. service request response time) ?

Real-world examples

- All the previos examples work with the notions of
  - service
  - service-oriented architecture
- The adaptation is expressed in terms of
  - service interface
  - architecture reconfiguration
    - changing the way the clients use the service

Services

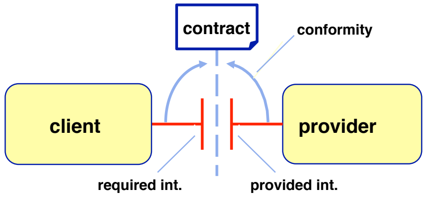
- Definition
  - A software system is a set of cooperating software components
  - “A service is a contractually defined behavior that can be implemented and provided by any component for use by any component, based solely on the contract” *

** Bieber and Carpenter, Introduction to Service-Oriented Programming, http://www.openwings.org*

Services and interfaces

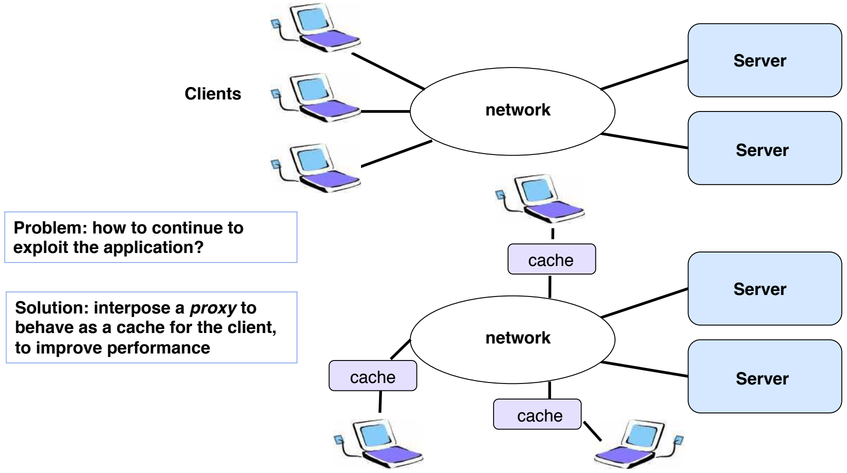
- Implementation
  - A service is accessible via one or multiple interfaces
  - An interface describes the interaction between serice povider and service client
    - Operational point of view:
      define operations and data structures for service implementation
    - Contractual point of view:
      define contract between service provider and service customer
  - Adaptation may be applied at both levels

Interface definition



- A service involves two interfaces
  - Required interface (from client side)
  - Provided interface (from provider side)

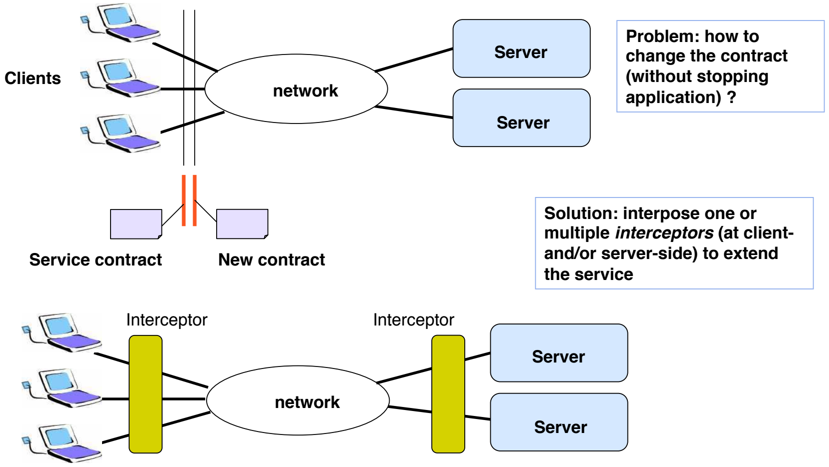
Ad-hoc adaptation – Interceptors Example 2: Service adaptation in case of mobility



Problem: how to continue to exploit the application?

Solution: interpose a *proxy* to behave as a cache for the client, to improve performance

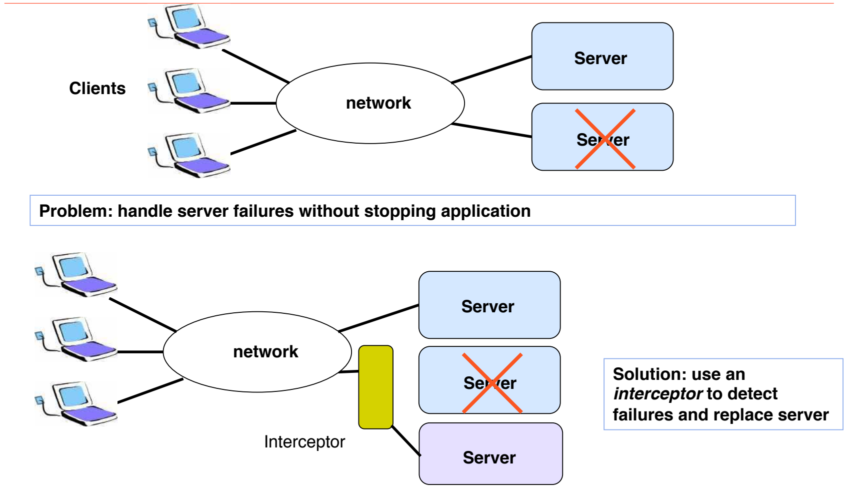
Ad-hoc adaptation – Interceptors Example 3: Service extension, evolution



Problem: how to change the contract (without stopping application) ?

Solution: interpose one or multiple *interceptors* (at client- and/or server-side) to extend the service

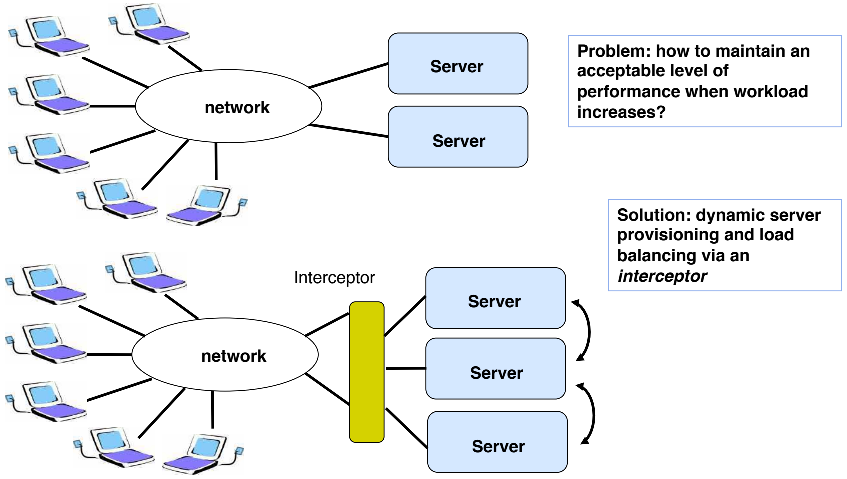
Ad-hoc adaptation – Interceptors Example 4: Service adaptation for fault tolerance



Problem: handle server failures without stopping application

Solution: use an *interceptor* to detect failures and replace server

Ad-hoc adaptation – Interceptors Example 5: Service adaptation for workload variation



Problem: how to maintain an acceptable level of performance when workload increases?

Solution: dynamic server provisioning and load balancing via an *interceptor*

Meta-object protocol (MOP)

- An adaptable service is organized in two levels
  - Base level
    - Implement functions defined by specifications
  - Meta-level
    - Use a representation of the base level to observe or modify its behavior
    - This meta-level representation is causally connected to the base level

The Java Case…

- Java uses reflection
  - The JVM represents classes using objects
  - All the information about classes and running objects may be consulted
    - The JVM provides the mechanisms to describe itself
    - this is called Java reflection

- The following tutorial will be used as a starting point in our lab
  - http://docs.oracle.com/javase/tutorial/reflect/

An example of acquiring information…

```
public class ClassSpy {
    public static void main(String... args) {
       try {
         Class<?> c = Class.forName(args[0]);
         out.format("Class:%n  %s%n%n", c.getCanonicalName());

          Package p = c.getPackage();
          out.format("Package:%n  %s%n%n",
                     (p != null ?
                      p.getName() :
                     "-- No Package --"));


       …
```

```
for (int i = 1; i < args.length; i++) {
  switch (ClassMember.valueOf(args[i])) {
  case CONSTRUCTOR:
       printMembers(c.getConstructors(), "Constructor");
       break;
  case FIELD:
       printMembers(c.getFields(), "Fields");
       break;
  case METHOD:
       printMembers(c.getMethods(), "Methods");
       break;
  case CLASS:
       printClasses(c);
       break;
  case ALL:
       printMembers(c.getConstructors(), "Constuctors");
       printMembers(c.getFields(), "Fields");
       printMembers(c.getMethods(), "Methods");
       printClasses(c); …
```

Running the Example

```
$ java ClassSpy java.nio.channels.ReadableByteChannel METHOD
Class:
  java.nio.channels.ReadableByteChannel

Package:
  java.nio.channels

Methods:
  public abstract int java.nio.channels.ReadableByteChannel.read
    (java.nio.ByteBuffer) throws java.io.IOException
  public abstract void java.nio.channels.Channel.close() throws
    java.io.IOException
  public abstract boolean java.nio.channels.Channel.isOpen()
```

http://docs.oracle.com/javase/tutorial/reflect/class/classMembers.html

An example of acting using this information

```
try {
          Class<?> c = Class.forName(args[0]);
          Object t = c.newInstance();
               …
          Method[] allMethods = c.getDeclaredMethods();
          for (Method m : allMethods) {
               String mname = m.getName();
               if (!mname.startsWith("test")…

          out.format("invoking %s()%n", mname);
          try {
               Object o = m.invoke(t, new Locale(args[1], args[2], args[3]));
               out.format("%s() returned %b%n", mname, (Boolean) o);
…
```

Running the example

```
$ java Deet Deet ja JP JP
invoking testDeet()
Locale = Japanese (Japan,JP),
ISO Language Code = jpn
testDeet() returned true
```

http://docs.oracle.com/javase/tutorial/reflect/member/methodInvocation.html

Conclusion

- All systems need adaptation
  - Requirements evolve
  - Execution contexts evolve
- Adaptation may be done at different levels, with different mechanisms
  - Trade-off between genericity and performance
- In ACS you will have the chance to learn about classic means of adaptation
  - Through specific technologies

References

- Lecture based on materials from
  - Sara Bouchenak, http://lig-membres.imag.fr/bouchenak/
  - Vijay Karamcheti, http://www.cs.nyu.edu/courses/fall99/G22.3033-003/index.htm
  - Sacha Krakowiak, http://sardes.inrialpes.fr/people/krakowia/