Conception d'Applications Réparties Avancée

CARA M2 Miage FA-FC

Jean-François Roos - bâtiment M3 extension - bureau 218 - Jean-Francois.Roos@univ-lille1.fr

2 janvier 2016

Sommaire

Plan du cours

- Introduction/rappel : notion de midleware
- Java Enterprise Edition : les composants serveurs
- Message Oriented Midleware : illustration avec JMS
- Sécurité dans JEE
- les Web Services

Objectifs du cours

- Approfondir la conception d'applications réparties à base d'objets et de composants
  - motivations et concepts
  - architectures et exemples
  - problèmes et solutions
- Comprendre les solutions industrielles
  - Java Entreprise Edition
  - Web Services
- Maîtriser par la pratique

Organisation

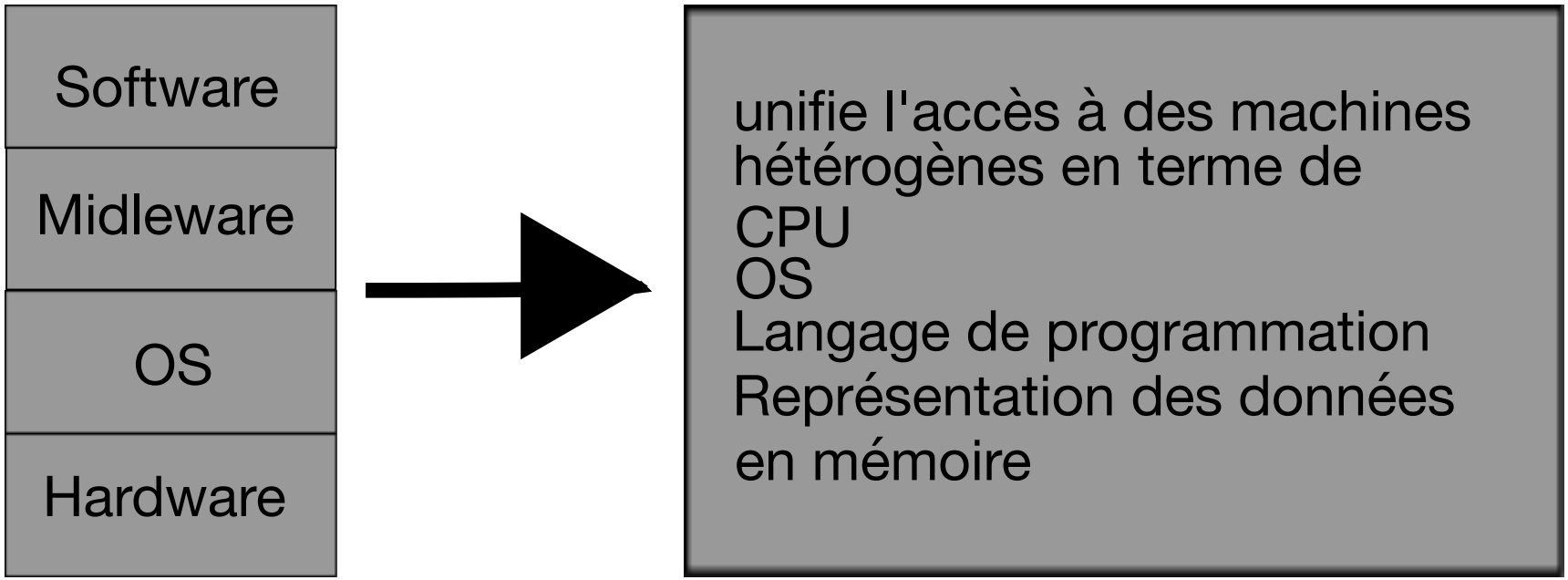
- Un cours et/ou un TD et/ou un TP par semaine
- Un même sujet de TP sera enrichi au fur et à mesure de la présentation de nouveaux concepts
- Un examen écrit à la fin du trimestre

Sommaire

Rappel

Midleware

Désigne dans le cadre de l'informatique répartie, toutes les couches logicielles qui permettent de communiquer à distance.



Objectif et positionnement

- Permettre à un programme de s'exécuter sur plusieurs machines reliées par un réseau
  - à large échelle (Internet)
  - local (intranet)
- Midleware : $\cap$ de plusieurs domaines de l'informatique
  - système d'exploitation - systèmes d'exploitation répartis
  - réseau - librairies de programmation réseau
  - langages de programmation - langages de programmation étendus

Fonctions

- masque l'hétérogénéité des machines et des systèmes
- masque la répartition des traitements et des données
- fournit une interface aux applications (modèle de programmation + API)
- permets à $\neq$ types de matériels (PC, mainframes, laptop, PDA, téléphones, ...) de communiquer à distance

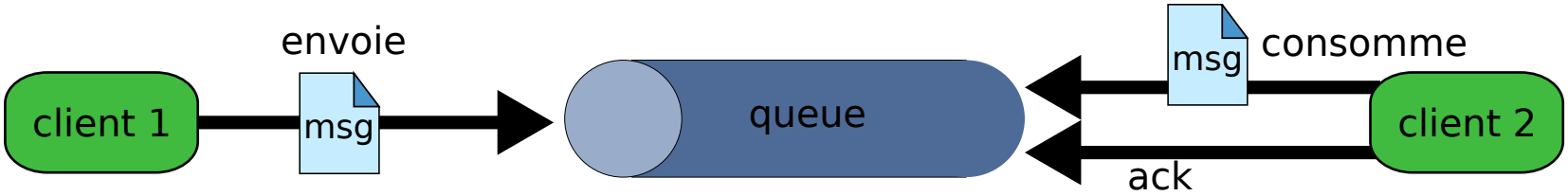
Paradigmes de communication : client/serveur

- le principal : interaction requête/réponse ou client/serveur
- une requête plus une réponse
- demande d'exécution d'un traitement à distance et réponse
- appel procédural étendu au cas où appelant et appelé ne sont pas situés sur la même machine
- communication synchrone

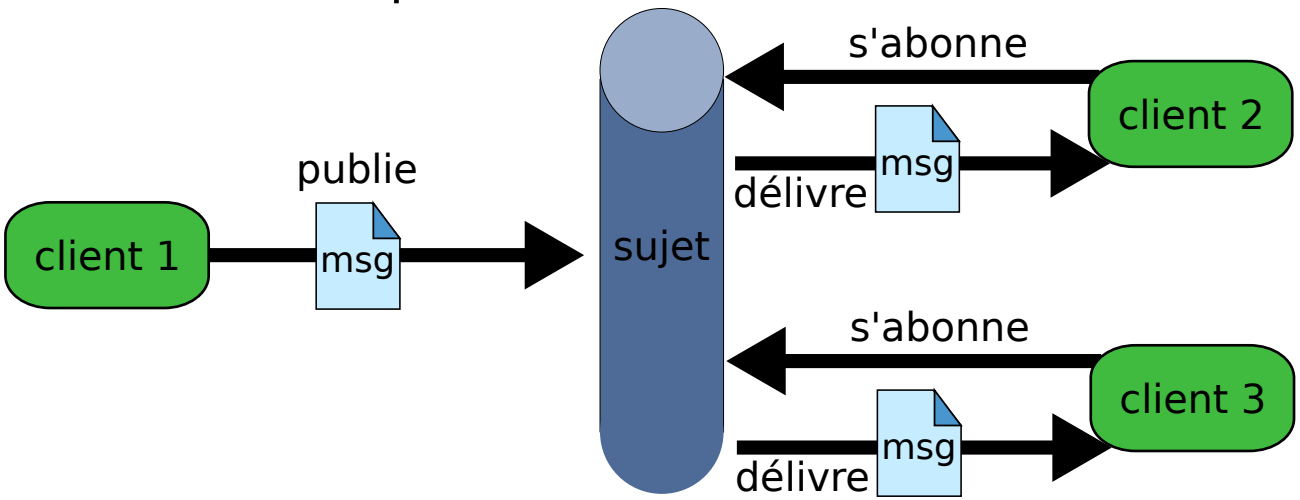
Paradigmes de communication : échanges de messages

- MOM (Message Oriented Middleware)
- envoi de messages différent de l'utilisation des sockets : ici protocoles applicatifs plus des propriétés par exemple transactionnelles
- communication asynchrone

- communication point à point



- communication par abonnement



Notion d'application client/serveur 2 tiers ou 3 tiers

Découpage d'une application en terme de :

- présentation
- traitements
- données

Problématique : par rapport à 1 client et 1 (ou +sieurs) serveurs qui assure ces fonctionnalités ?

Client/Serveur 2 tiers

Client : présentation
Serveur : données + traitement
Caractéristiques :

- 1 gros serveur (mainframes)
- n terminaux légers connectés au serveur

Avantages :

- pas de duplication de données (état global observable)
- gestion simple de la cohérence et de l'intégrité des données
- maîtrise globale des traitements

Inconvénients :

- modèle trop rigide qui n'assure pas l'évolutivité souvent
- solutions propriétaires fermés économiquement trop coûteux

Client/Serveur 3 tiers

Client : présentation
Un Serveur : traitement
Un Serveur : données
Caractéristiques :

- 1 serveur de données et un serveur de traitement
- n PC avec IHM évoluées

Avantages :

- meilleure répartition de charge
- économiquement moins cher
- plus évolutif

Inconvénients :

- administration plus compliquée
- mise en œuvre plus compliquée

Évolution historique du terme client/serveur 3 tiers

tiers 1 (PC) tiers 2 (serveurs départementaux) tiers 3 (serveur central)

tiers 1 (client) tiers 2 (BD locale) tiers 3 (BD globale)

Actuellement

tiers 1 (client) tiers 2 (serveur traitement) tiers 3 (serveur données)

Évolution du midleware

- envoi de message
- RPC
- RPC objet
- bus logiciel
- serveur d'applications
- service

Envoi de messages

- primitives send & receive
- la conception des programmes client et serveur est fonction des messages attendus et à envoyer

- socket : au-dessus des protocoles TCP & UDP
- primitive bloquante vs non bloquante
- fiabilité
- ordre des messages
- contrôle de flux
- mode connecté vs non connecté

Remote Procedure Call (RPC)

- appel d'une procédure sur une machine distante
- groupement de 2 messages : appel & retour
- adressage : @IP + nom fonction
- définition des signatures des procédures
- compilateur de souches client et serveur

Exemple : RPC Sun

```
struct bichaine { char s1[80]; char s2[80]; };

program CALCUL {
  version V1 {
    int multpar2(int) = 1;
    string concat(struct bichaine) = 2;
    void plus_un() = 3;
  } = 1;
} = 0x21234567;
```

RPC Objet

- mise en commun concepts RPC et programmation objet
- appel d'une méthode sur un objet distant
- éventuellement plusieurs objets par machine
- adressage serveur de noms + nom logique

Exemple : Java RMI

```
import java.rmi.Remote;
import java.rmi.RemoteException;

interface CompteInterf extends Remote {
  public String getTitulaire() throws RemoteException;
  public float solde() throws RemoteException;
  public void deposer(float montant) throws RemoteException;
  public void retirer(float montant) throws RemoteException;
  public List historique() throws RemoteException;
}
```

Bus logiciel CORBA

- multi-OS, multi-langage
- métaphore du bus logiciel
- langage IDL

```
module MyApp {
  interface CalculatriceItf {
    double add( in double val1, in double val2 );
    double sub( in double val1, in double val2 );
    double mult( in double val1, in double val2 );
    double div( in double val1, in double val2 );
  };
};
```

- mappings (traduction) vers langages : Java, C++, C, Ada, COBOL, ...
- tout est objet (y compris services fournis par le bus : annuaire, ...)

Composants et serveurs d'application

- CORBA CCM
- Sun EJB
- Microsoft .NET/(D)COM+

composant :

- monter en abstraction / objet
- architecture logicielle

serveur d'applications :

- framework
- héberger, administrer

Service

- SOA : Software Oriented Architecture
- SaaS : Software As A Service

Modèle «économique» : tout est service
Principes des SOA :

- Encapsulation existant encapsulé pour pouvoir être réutilisé avec SOA
- Loose Coupling minimiser les dépenses
- Contract communication ne se font que via un accord entre services
- Abstraction masquer la logique du service au monde extérieur
- Reusability découpage des services pour promouvoir la réutilisabilité
- Composability services peuvent être composés et coordonnés pour former des services composites
- Autonomy services contrôlent la logique qu'ils encapsulent
- Optimization services peuvent être optimisés individuellement
- Discoverability services sont faits pour être découvert

Service 1ère vague Épuration

Constat d'une trop grande hétérogénéité du middleware
taille, OS, langage, cible matérielle, modèle de programmation

Focalisation sur un PPCM (plus pétit commun dénominateur)
d'où W3C Web Services = HTTP + XML + SOAP

adoption par Sun (Java EE), Microsoft (.NET) comme solution pour l'intéropérabilité (interne et externe)

Service 2ème vague ... en fait

réintroduction des notions de composants et d'architecture logicielle

- Enterprise Software Bus (par ex. Sun JBI)
- OSGi
  - domotique (box, ...)
  - embarqué (secteur automobile, ...)
  - modularité, système à plugins (Eclipse, serveur d'applications (Glassfish))
- SCA (Software Component Architecture)
  - donner une structuration aux applications orientée services
  - supporter différents langages de programmation, protocoles de communication, langages de définition d'interfaces, services non fonctionnels

- objet, composant, service
- interface, contrat
- souche, squelette
- service techniques (aussi appelés non fonctionnels ou extra-fonctionnels)
- annuaires (registre, moteur de recherche, pages jaunes, ...)
- sécurité (contrôle d'accès, cryptage, authentification, ...)
- transaction
- persistence des données
- concurrence
- synchronisation
- réplication
- migration
- ...

Conclusion

Fournisseurs de solutions middleware

- OMG CORBA
- W3C Web Services
- OSGi service+composant+architecture logicielle (Java)
- OSOA service+composant+architecture logicielle (Java, C++, PHP, ...)
- Oracle Java+Java EE (JEE)
  IBM, BEA, ... fournisseurs de solutions Java EE
- Microsoft C# + .NET

mais aussi tous les autres grands du domaine (HP,BEA,IONA,Fujitsu,SAP, ...)
de nombreuses briques open-source (Apache, Eclipse, OW2/ObjectWeb)
de nombreux domaines applicatifs :

- applications Web (commerce en ligne, ...), systèmes d'information, BD
- embarqué (domotique, applications mobiles téléphones, réseau de capteurs, ...)
- infrastructure télécom
- calcul intensif sur clusters et grilles