# Cours 3 : Applications réparties

Guillaume LOUP
Master 2 (ISC - TNI)
*2020-2021*

*Supports de M. Chellali*

https://ecampus.paris-saclay.fr/course/view.php?id=11300



## IPoAC

RFC 1149
1990 : description par David Waitzman

RFC 2549
1999 : intégration de « qualité de service »

RFC 6214
2011 : adaptation à l'IPv6





## IP over Avian Carriers

*IP par transporteurs aviaires*

Test 2001

Envoi de 9 paquets sur 5kms par un groupe d'utilisateurs Linux de Norvège

Chaque paquet est porté par un seul pigeon est fait office de ping

Résultats : 55% des paquets perdus avec un temps de réponse compris entre 3000 et 6000s





## IP over Avian Carriers

*IP par transporteurs aviaires*

Test 2009

Envoi de 4 Go sur 96kms par une seule mémoire flash en Afrique du Sud

2h pour le pigeon
25h en ADSL

# Plan

- Introduction
- Architectures
- Systèmes distribués
- Distribution des données
- Distribution des traitements
- Cloud Computing
- Conclusion





## Pourquoi des applications réparties ?

- Besoins propres des applications
  - Intégration d'applications existantes initialement séparées
  - Intégration massive de ressources
    - Grilles de calcul, gestion de données
  - Pénétration de l'informatique dans des domaines nouveaux d'application
    - Intégration d'objets du monde réel (informatique omniprésente (ubiquitous computing))





## Pourquoi des applications réparties ?

- Possibilités techniques
  - Coût et performances des machines et des communications
  - Interconnexion généralisée
    - Exemple 1 : interpénétration informatique-télécom-télévision
    - Exemple 2 : réseaux de capteurs





- Introduction

# Architectures

- Systèmes distribués
- Distribution des données
- Distribution des traitements
- Cloud Computing
- Conclusion

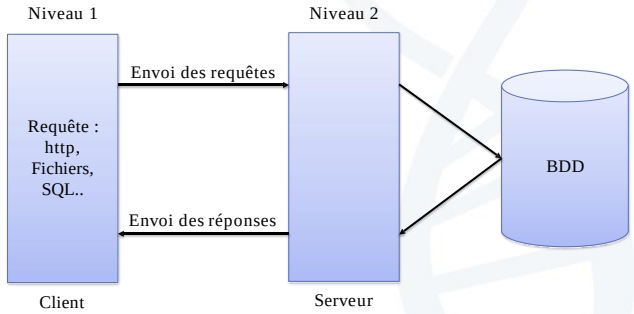
# Architecture 2-tiers (client-serveur)

- Application décomposée en clients et services
- Les services :
  - Sont passifs (attendent une connexion)
  - Réceptionnent une requête
  - Envoient une réponse
- Les clients
  - Sont actifs : initient une connexion
  - Envoient des requêtes
  - Attendent les réponses des services
  - Reprennent leur exécution
- La communication est synchrone (dans le modèle de base) : le client est bloqué en attente de la réponse





# Architecture 2-tiers (client-serveur)







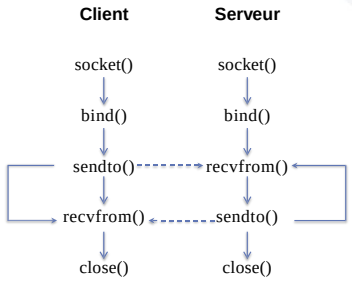

# Rappel : Modes de transport

- TCP : mode connecté
  - Fiable mais consommateur de bande passante
  - Contrôle des informations envoyées
  - Réémission des paquets perdus
  - Elimination des paquets dupliqués
  - Adaptation du débit
- UDP : mode non connecté
  - Communication unidirectionnelle simple
  - Utilisé pour le streaming
  - Utilisé dans les réseaux considérés fiables
  - Peu consommateur en bande passante



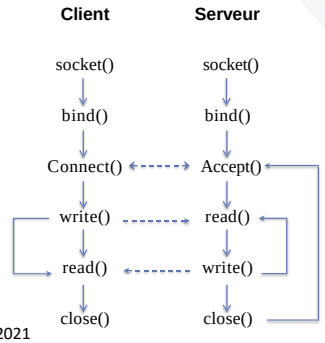


# Client/serveur UDP




- Rôle ambivalent du client et du serveur
- Echanges simples : communication unilatérales
- Peu d'appels systèmes en définitive

# Client/serveur TCP




- Rôle séparés
- Communication bilatérales
- Appels systèmes plus nombreux





# Architecture 2-tiers : inconvénients

- Un client "lourd"
  - Contient toute la logique de présentation des données
  - Interagit directement avec le serveur de données
- Problèmes :
  - Inadapté pour de grandes structures
  - Déploiement du client lourd sur un parc de machines
    - Maintenance longue et coûteuse !
  - Effort de développement généralement important
    - Nécessite des connaissances en IHM
    - Nécessite des connaissances avancées en réseaux





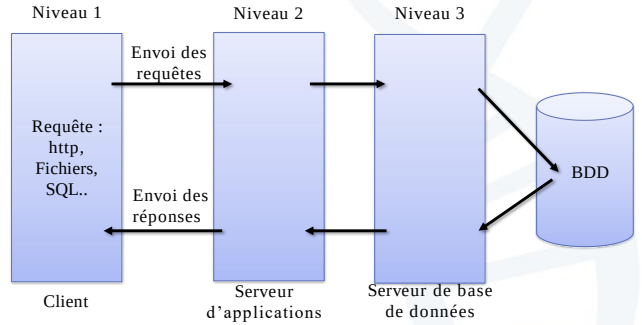
# Architecture 3-tiers

- Principe
  - Utiliser un client léger (par ex. navigateur web)
  - Centraliser la logique applicative sur un serveur
    - Déporter la logique de traitement sur un serveur
- Organisation en 3 couches
  - La couche Présentation
  - La couche Métier
  - La couche Accès aux données

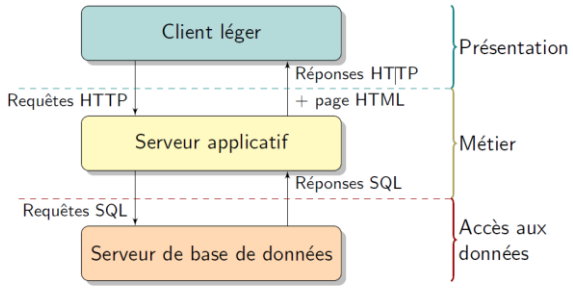




# Architecture 3-tiers



# Architecture 3-tiers






# Couche présentation

- Partie visible et interactive de l'application (IHM)
- Affiche et restitue les données sur le client
- Dialogue avec l'utilisateur :
  - Relaie les requêtes utilisateur à la couche métier
  - Présente les informations retournées par cette dernière
- Associée au client léger
  - Navigateur internet
  - Guichet et distributeurs automatiques





# Couche présentation

- Peut avoir des représentations différentes sans changer la finalité de l'application
  - Application graphique (WIMP) ou textuelle (lignes de commande)
  - Représentée en HTML (navigateur web) ou en WML (Smartphone, tablette)
- Exemple : distributeur de billet
  - Représentations différentes (selon la banque, le pays...etc.)
  - Les fonctionnalités et services restes identiques





# Couche Métier (Business)

- Parfois aussi appelée Middleware
- Partie fonctionnelle de l'application
  - Implémente la logique de traitement de l'application
  - Décrit les opérations sur les données
- Elle ne change pas si on change l'interface utilisateur ou la façon d'accéder aux données

# Couche Métier (Business)

- Offre des services applicatifs et métier à la couche Présentation et affiche les résultats des traitements
- S'appuie sur les services fournis par la couche Accès aux données
- Met en œuvre les règles de gestion et de contrôle du système





## Couche Accès aux données (data access)

- Gère l'accès aux données du système, voire de celles d'autres systèmes
- Fournit des services d'accès aux données à la couche métier
- Service encapsulant une base de données (cas général)





## Couche Accès aux données (data access)

- Données propres à l'application
  - Données pérennes
  - Stockées dans de simples fichiers (XML ou binaires) ou dans une base de données
  - Accès aux données est le même quelque soit le support
- Données gérées par une autre application
  - L'application s'appuie sur la capacité d'une autre application à fournir des données
  - Données volatiles





# Avantages

- Allègement du poste de travail client
- Prise en compte de l'hétérogénéité des plates-formes
- Amélioration de la sécurité des données, en supprimant le lien entre le client et les données
- Rupture du lien de propriété exclusive entre application et données
- Meilleure répartition de la charge entre différents serveurs d'applications

## Déclinaison des architectures multi-tiers

- Architecture 2-tiers : client-serveur classique
- Architecture 3-tiers : application web courante
  - Client léger (par ex. navigateur Web)
  - Un serveur d'application (par ex. serveur Web)
  - Un serveur de base de données
- Et si on généralise ?





## Architecture multi-tiers

Une généralisation :

- Multiplication des services
- Partage de l'application ou des ressources entre ces services
  - On parle alors de distribution ou répartition





## Architecture multi-tiers

- Architecture multi-tiers :
  - Architecture client**s**-serveur**s**
  - Application exécutée par plusieurs agents logiciels
  - "Tier" veut dire classé/disposé/rangé en anglais
  - Français : Architecture multi-niveau





- Introduction
- Architectures

## Systèmes distribués

- Distribution des données
- Distribution des traitements
- Cloud Computing
- Conclusion

## Définition des applications réparties

- Application qui, suivant les principes de l'architecture client-serveur, peut tourner de façon transparente sur plusieurs ordinateurs reliés à travers un réseau informatique, indépendamment du système utilisé
  - Plusieurs services mais pas forcement plusieurs serveurs
  - Fait surtout appel à un ensemble de techniques de programmation
- Synonymes
  - Applications n-tiers, applications **distribués**





## Définition d'un système réparti

- Ensemble composé d'éléments reliés par un système de communication
- Ces éléments ont des fonctions de traitement (**processeurs**), de stockage (**mémoire**), de relation avec le monde extérieur (**capteurs**, **actionneurs**)





## Systèmes réparties

- Les différents éléments du système ne fonctionnent pas indépendamment mais collaborent pour réaliser une ou plusieurs tâches communes
- Conséquence :une partie au moins de l'état global du système est partagée entre plusieurs éléments (sinon, on aurait un fonctionnement indépendant)





## Caractéristiques des systèmes répartis

- Propriétés souhaitées
  - Le système doit pouvoir fonctionner (au moins de façon dégradée) même en cas de défaillance de certains de ses éléments
  - Le système doit pouvoir résister à des perturbations du système de communication (perte de messages, déconnexion temporaire, performances dégradées)
  - Le système doit pouvoir résister à des attaques contre sa sécurité (violation de la confidentialité, de l'intégrité, usage indu de ressources, déni de service)

## Caractéristiques des systèmes répartis

- **Conséquences**
  - Des décisions doivent pouvoir être prises localement, et dans une situation d'incertitude, (sans connaissance d'un état global, d'ailleurs difficile à définir)





## Difficultés

- Propriété d'asynchronisme du système de communication (pas de borne supérieure stricte pour le temps de transmission d'un message)
- **Conséquence** : difficulté pour détecter les défaillances





## Difficultés

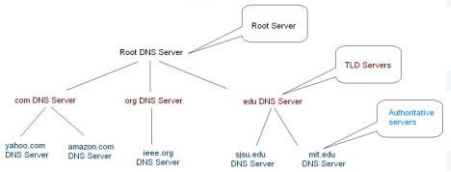
- Dynamisme (la composition du système change en permanence)
- **Conséquences :**
  - Difficulté pour définir un état global
  - Difficulté pour administrer le système





## Difficultés

- Malgré ces difficultés, des grands systèmes répartis existent et sont largement utilisés
  - le DNS (Domain Name System)
    - Adresse IP des sites web difficiles à mémoriser pour les utilisateurs
    - Adresse alphanumérique FQDN : Fully Qualified Domain Name
    - Le système DNS assure le lien entre les deux types d'adresse.



## Qu'est ce qui peut être reparti ?

- Des données : base de données distribuées
  - Données reparties sur plusieurs machines reliées en réseau
- Des traitements
  - Des fonctions exécutées à distance
  - Des objets distants
  - Des tâches en parallèle sur un parc homogène ou hétérogène de machines :
    - Lié aux notions de cluster, grid et cloud computing
- Idée générale
  - Plusieurs machines font plus de choses qu'une seule!





- Introduction
- Architectures
- Systèmes distribués

## Distribution des données

- Distribution des traitements
- Cloud Computing
- Conclusion





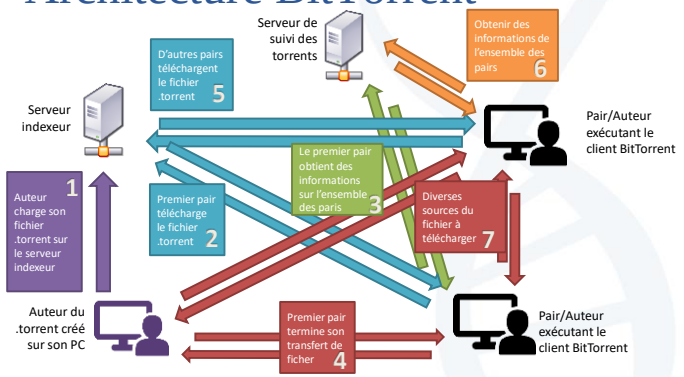
## Distribution des données

- Quand les données sont sur plusieurs machines...
  - Exemple 1 : base de données distribuées
    - DNS : Domain Name System (résolution des noms de domaine)
  - Exemple 2 : distribution de fichiers
- Protocoles d'échange de données de type peer-to-peer :
  - Bitcoin
  - µTorrent





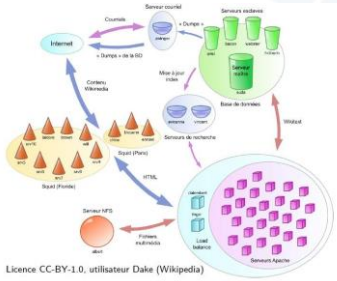
## Architecture BitTorrent




*Borges, R., Patton, R. M., Kettani, H., & Masalmah, Y. (2011). Bitpredator: A discovery algorithm for bittorrent initial seeders and peers. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).*

# Distribution des traitements

- Répartition de la charge
- Exemple : wikipedia : 40 000 requêtes/s









# Distribution des traitements

- Calcul scientifique massivement parallèle
- Type de calcul
  - Simulations d'écoulement de fluides
  - Simulations nucléaires, . . .
- Matériel
  - Utilisation de clusters de machines
  - Utilisation de super-calculateurs
- Outils de programmation
  - Abstractions de machines en réseau et/ou multi-processeurs :
    - PVM (Parallel Virtual Machine)
    - MPI (Message Passing Interface)





# Distribution des traitements

- Plateformes de calcul distribué (grid computing)
  - Exemple : Folding@home
  - Etude du repliement des protéines pour comprendre les mécanismes liés au cancer, à la maladie d'Alzheimer, au fonctionnement des médicaments
  - http://folding.stanford.edu





# Noms de domaines

Problème :

- Difficulté à retenir les adresses IP
  - Se rappeler d'une adresse IP est assez difficile : 212.27.48.10
  - Mais alors de plusieurs (209.85.137.99, 139.124.187.4,... ), c'est franchement pénible !
- Retenir des noms comme www.free.fr, www.google.com, www.facebook.fr : plus facile
- Nécessite d'utiliser un système plus parlant !
- Création d'un système de nom de plus haut niveau : DNS
  - Annuaire pour Internet

# Domain Name System : DNS

- Résolution de nom = la relation entre un nom de machine (hostname) et son adresse IP
- Résoudre un nom de domaine consiste à trouver l'adresse IP qui lui est associée
- La correspondance entre un nom de machine et une adresse IP est bien normalisée
- Pour que ce système fonctionne, il faut qu'il y ait au plus une adresse IP qui corresponde au nom d'une machine
- Google.fr : 193.51.224.177





# Noms de domaines

- Organisation hiérarchique des réseaux, donc des noms (regroupement en domaines)
- Noms appelés aussi DNS (Domain Name System), 1987 (RFC 1034, 1035)
- Base de donnée des noms distribuée :
  - Chaque domaine possède un ou plusieurs contrôleurs de domaine (DNS) stockant une partie de la BDD





# Hiérarchie des domaines

- Hiérarchie arborescente (nœuds + feuille)
- Un nœud = un domaine :
  - Un ou plusieurs contrôleurs de domaine par nœud
  - Chaque nœud porte un nom, sauf la racine
- Une feuille = une machine :
  - Nom complet donné en remontant de la feuille vers la racine
  - Séparateur entre chaque domaine : .

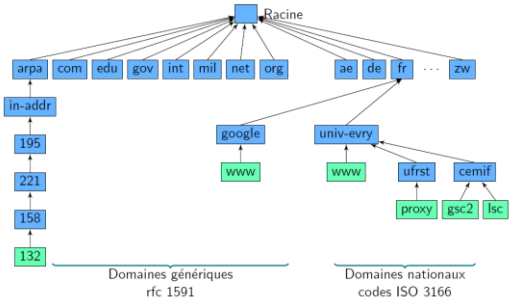




# Convention de nommage

- Insensibilité à la casse
- Chiffres, - et _ autorisés, espaces et tabulations interdits
- 63 caractères max par nœuds, 12 par convention
- Nom complet : 255 caractères max

# Structure générale






# Le contrôleur de domaine

- Rôles
  - Gérer les noms d'un domaine
  - Répondre aux requêtes DNS
  - Mettre en cache des informations DNS
- Informations stockées
  - Couples noms/adresses IP des machines de son domaine
  - Couples domaines/adresses IP des contrôleurs des domaines pères
  - Couples sous-domaines/adresses IP des contrôleurs de sous-domaine





# Le contrôleur de domaine

Conséquences

- Un contrôleur de domaine ne possède pas toutes les correspondances adresses IP/nom de machine du monde
  - Données reparties sur les différents contrôleurs de domaine
  - Retrouver l'adresse IP associée à un nom de machine peut faire intervenir plusieurs contrôleurs de domaines





# Client DNS

Rôle

- Récupérer les correspondances nom de machine/adresse IP auprès de ses serveurs DNS
  - Généralement, serveur DNS = contrôleur de domaine
- Mettre en cache les informations DNS

# Client DNS

Modes d'interrogation

- itératif : le client interroge tour à tour les contrôleurs de domaine en commençant par son serveur DNS
- récursif : le client interroge son serveur DNS qui interroge tour à tour les contrôleurs de domaine (client plus simple à programmer)





# Interrogations successives

Principe (en mode récursif)

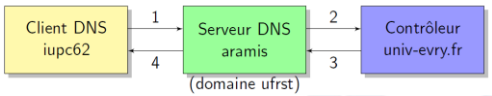
1. Le client DNS interroge le serveur DNS
2. Si le serveur DNS ne connaît pas la correspondance adresse IP/nom (sinon aller en 4) :
   - Identifie le domaine père commun
   - Interroge le contrôleur de domaine concerné
   - Ce dernier indique le contrôleur de domaine fils concerné
3. Le serveur DNS interroge les contrôleurs de domaine de fils en fils jusqu'à obtenir la correspondance adresse IP/nom
4. Le serveur DNS retourne la réponse au client DNS





# Exemple 1

- iupc62.ufrst.univ-evry.fr veut contacter www.univ-evry.fr et a pour serveur DNS : aramis.ufrst.univ-evry.fr qui gère le domaine ufrst

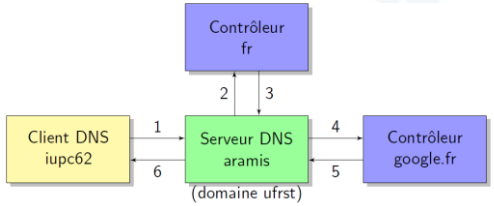







# Exemple 2

- iupc62.ufrst.univ-evry.fr veut contacter www.google.fr



# Implémentation du DNS

- Protocoles de transport
  - UDP (port 53), 520 octets max :
    - Pour requêtes standards
  - TCP (port 53)
    - Pour échanges de zones de contrôleurs à contrôleurs
- Formats imposés
  - Pour les datagrammes de requête
  - Pour les données dans la base







# Distribution des traitements

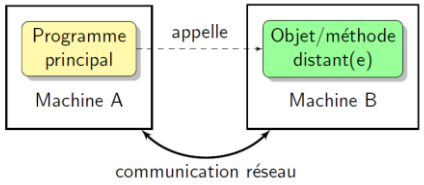






# Problématique

- Accès à des ressources de calcul distantes
- Une partie du programme est déportée sur une autre machine
- Les deux parties doivent pouvoir communiquer








# Solutions

- Solution 1 : implémentation à la main
  - Pour chaque situation, on programme tout à l'aide des sockets !
  - Solution fastidieuse à long terme !
- Solution 2 : mettre en place ou utiliser une solution générique
  - Solution qui doit fonctionner pour n'importe quel type d'objet
  - L'objectif est de s'affranchir de la programmation bas niveau des sockets !

# Solution générique

Côté programme principal

- On doit :
  - Disposer d'une description de l'objet distant
  - Connaître l'endroit ou il est localisé
  - Appeler les fonctions de cet objet

Côté objet distant

- On doit fonctionner comme un serveur
  - Réceptionner et interpréter les requêtes
  - Effectuer les traitements demandés
  - Retourner le résultat des traitements





# Principaux problèmes

- Description générique d'un objet
  - Affectation des ressources au matériel (localisation)
  - Comment encoder/décoder les informations de manière générique :
    - de manière à ce que l'objet distant sache :
      - la méthode à lancer
      - les paramètres à lui passer
    - de manière à ce que le programme principal :
      - interprète le résultat fourni
      - ou détecte un problème dans la transmission
- Quel protocole de communication mettre en place ?





# Interface

- Définition (au sens programmation orienté objet) :
  - description du comportement d'un composant logiciel
  - ensemble de méthodes sans implémentations
  - sorte de contrat passé entre :
    - un objet qui utilise l'interface (compte sur le service fourni)
    - un objet qui réalise l'interface (fourni les services décrits)





# Sérialisation et désérialisation

- Sérialisation (marshalling) : processus visant à encoder un objet en mémoire sous la forme d'une suite d'octets
- Désérialisation (unmarshalling) : processus inverse, qui à partir d'une suite d'octets, reconstitue les données

# Principe

- Une interface décrivant un objet distant
- Deux implémentations :
  - Une implémentation locale (la souche - stub) :
    - Sérialise les requêtes
    - Gère la communication avec l'objet distant
    - Desérialise les réponses
  - Une implémentation distante (le squelette - skeleton) :
    - Desérialise les requêtes
    - Lance les méthodes de l'objet ;
    - Sérialise la réponse





# Communication réseau

- Modèle en couche calqué sur le modèle OSI
  - Application (client/serveur) ;
  - Squelette/souche ;
  - Sessions générique ;
  - Transport (souvent TCP)






# RPC : Remote Procedure Call

- Standard de fait : SunRPC (RFC 1057)
  - Application : appel de fonction standard
- **Très nombreuses implantations** différentes du RPC ayant des sémantiques variées
- L'objectif affiché par la plupart des implantations serait pour le programmeur :
  - **de retrouver la sémantique habituelle** de l'appel de procédure en mode centralisé
  - **sans se préoccuper de la localisation de la procédure exécutée** (sauf s'il le souhaite explicitement)





## CORBA : Common Object Request Broker Architecture

- Mécanisme indépendant du langage de programmation
  - Application : objet distant
  - Squelette/souche : langage IDL CORBA
  - Session :
    - Utilisation du bus CORBA nommé IIOP (Internet Inter-ORB Protocol) qui repose sur TCP/IP
    - Lancement d'un service réseau
    - Utilisation de l'API CORBA

## JSON-RPC : RPC avec JavaScript Object Notation

Orienté web service

- Application : service web distant avec méthodes
- Squelette/souche : néant
- Session : session HTTP
- Format de communication : JSON
- Et beaucoup d'autres ...
  - XML-RPC, SOAP, Thrift (Facebook), Microsoft DCOM, Microsoft WCF (.Net), . . .



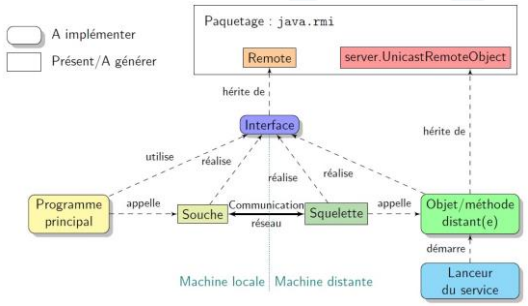


## RMI : Remote Method Invocation

- Un mécanisme propre à Java
- Plus simple à mettre en œuvre que CORBA :
  - Application :
    - objet distant : hérite de java.rmi.server.UnicastRemoteObject
    - Interface : hérite de java.rmi.Remote
  - Squelettes et souches :
    - Créés avec rmic (optionnel)
  - Session :
    - Lancement du service « registre RMI » et du serveur
    - Client employant les primitives Naming.lookup().



## RMI






- Introduction
- Architectures
- Systèmes distribués
- Distribution des données
- Distribution des traitements

## Cloud Computing

- Conclusion

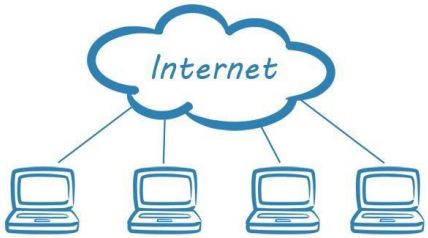
# Introduction

- Transformations de fond du processus informatique :
  - Temps partagé (matériel)
  - Ordinateur personnel (logiciel)
  - En réseau (information)
  - Dans les nuages (service)
- De tout temps, l'internet a été représenté comme un nuage :
  - On ne regarde pas ce qui s'y passe, mais juste les effets (connections)





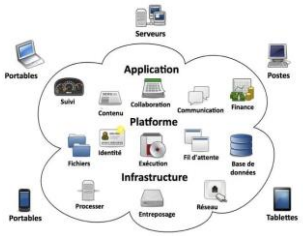
# Introduction






# Introduction

- Maintenant, ce sont les serveurs qui intègrent ce nuage




Le Nuage





# Définitions

- JO : "Mode de traitement des données d'un client, dont l'exploitation s'effectue par l'Internet, sous la forme de services fournis par un prestataire

  Note : L'informatique en nuage est une forme particulière de gérance de l'informatique, dans laquelle l'emplacement et le fonctionnement du nuage ne sont pas portés à la connaissance des clients."

# Définitions

- *Le cloud computing est l'accès via le réseau, à la demande et en libre-service, à des ressources informatiques virtualisées et mutualisée* (NIST)





# Définitions

- Informatique dans les nuages, informatique délocalisée
- Concept de déportation sur des serveurs distants des traitements informatiques traditionnellement localisés sur le poste utilisateur





# Besoins

- Internet côté "grand public" :
  - messagerie classique et instantanée
  - moteurs de recherche, accès direct à la mémoire du web
  - consultation de comptes (banques, assurances, sante, éducation)
  - réseaux sociaux et partage d'information (news, photos, ...)
  - consultation de catalogue (vente en ligne, enchères)
- Problèmes utilisateur : manque de formation, gestion des accès confidentiels (gestion de mots de passe et des numéros d'accès), peur du hacking et du mauvais usage, respect de la vie privée
- Point positif (+++) : services assurés instantanément (ou presque)





# Besoins

- Internet Côté entreprise :
  - travail collaboratif et simultané sur un même document
  - traduction simultanée des appels
  - bureau mobile
  - tout à la fois
- Pas encore complètement réalité mais besoin accentué durant la crise sanitaire
- Nécessite d'utiliser une infrastructure de type Internet
- Applications très puissantes et réactives (lesquelles ? placées où ?)

# Besoins

- Internet Côté entreprise :
  - usage ponctuel de software avec licence
  - gestion de gros volumes de données
  - disponibilité 7j/7 et 24h/24 de ses données et des services
- Payer uniquement un usage temporaire
  - Quelles infrastructure ? Depuis quelle plateforme ?
- Problème très important : la confidentialité





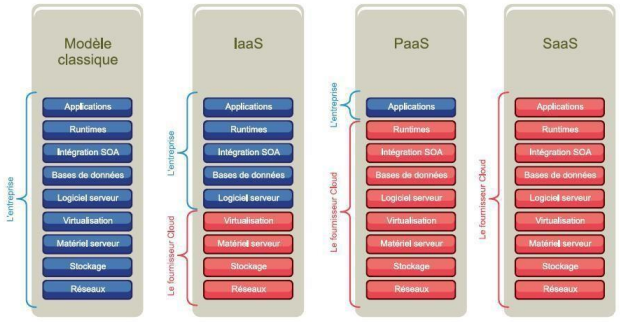
# Trois modèles fondamentaux

- **Infrastructure as a Service** (IaaS )
- **Plateform as a Service** (PaaS )
- **Software as a Service** (Saas)
- Le degré d'externalisation est variable
- Les concepts de IaaS, PaaS, et Saas indiquent ce degré

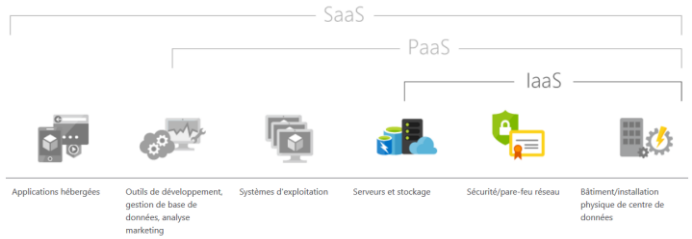


# Trois modèles fondamentaux

*IaaS, PaaS, SaaS : qui maintient quoi ?*




# Trois modèles fondamentaux



# IaaS

- Infrastructure as a Service
- Disposer d'une infrastructure (serveurs, stockage, réseau) hébergée
- L'accès à la ressource est complet et sans restriction, équivalent de fait à la mise à disposition d'une infrastructure physique réelle





# IaaS

- Virtualisation
- Offre de capacité de calcul "élastique"
  - Exemples :
    - Amazon EC2
    - Windows Azure





# PaaS

- Platform as a Service
- Désigne qu'on dispose d'une plateforme capable d'accueillir les applications de l'entreprise et tous les environnements et outils de gestion et de test
- L'environnement est prêt à l'emploi, fonctionnel et performant, y compris en production
- L'infrastructure hébergée est totalement transparente





# SaaS

- Software as a Service
- Désigne la déportation des applications de l'entreprise dans le cloud
- Les Cloud Computing Saas proposent des logiciels opérationnels prêt à l'emploi sans aucune installation ou opération de maintenance
- SaaS signifie qu'on vend à l'entreprise un service de traitement des données au sein de l'entreprise. On parle d'operateur de service (et non plus d'éditeur de logiciel) pour designer le fournisseur SaaS

# SaaS

- Logiciel installé à distance sur le serveur
- Utilisable par plusieurs en même temps
- Mise à jour sous la responsabilité du serveur
- Les documents restent accessibles par export... mais restent également sur le serveur
- Redondance et sécurité (de l'information, pas de l'usager)





# Applications SaaS

- **Documents partagés :**
  - Écriture collective
  - Accès depuis de nombreux terminaux
    - Etherpad (libre)
    - Google Drive
    - Microsoft Office 365
  - Lecture sur des appareils hétérogènes (tablettes...)
- **Mail**
  - Accès au travers du web (webmail)
  - Carnet d'adresse dans les nuages
    - Que des applications tierces peuvent utiliser (inscription à G+)
  - Publicité ciblée dans le mail (personnel)
  - Service complémentaire, participant de la captation de clientèle





# Applications SaaS

- **Stockage à distance**
  - « Synchroniser »
  - Sauvegarde et moyen de partage
  - Multi-terminaux
    - Dropbox
    - Google Drive
    - Owncloud
  - Question de la confiance
- **Streaming personnel**
  - Ne plus posséder sa musique sur chaque appareil de lecture
  - Accès vs Téléchargement
    - Icloud
  - Licence légale « privée »
  - Vers la fin du partage ?





# Avantages

- Mise à jour des logiciels
- Le gestionnaire ayant accès aux données peut procéder à la mise à jour en cas de changement de format
- Le système connaît les points de blocages pour les améliorer
- Solutions pour les grands parcs de machines hétérogènes (entreprises)

# Inconvénients

- Données accessibles par l'opérateur de nuage (profilage vs service)
- Quelle juridiction de protection des données appliquer (Europe/ USA)
- Applications simplifiées pour être utilisables sur tous les terminaux (facilité vs créativité & performance)
- Perte de l'autonomie (facilité vs compréhension)





# Modèle de déploiement

- Private cloud
- Public cloud
  - Amazon
  - Google
  - Microsoft Azure
- Community cloud
  - Owncloud
  - Seafile
- Nuages de service public
  - Projet français





# Risques entreprise

- Confidentialité (PME / grande entreprise)
- Conformité réglementaire
- Rejet de la part des partenaires (clients, fournisseurs)





# Point de vue utilisateurs

- Ergonomie et productivité des applications
- Accessibilité
- Collaboration
- Agilité
- Qualité de service et disponibilité
- Renouvellement des machines
- Dépossession du poste de travail
- Confidentialité des données

# Nouvelle architecture

- Internet
  - Horizontal
  - Code ouvert
  - Peu de points de contrôle, interactions
  - Robustes entre usagers
  - Partage (et téléchargement)
  - Autonomie des postes, des usagers
- Cloud
  - Structure hiérarchique
  - centralisée
  - Avec un point central de contrôle
  - Jardins privés (modèle vertical de Apple et Facebook)
  - Streaming
  - Clients légers





- Introduction
- Architectures
- Systèmes distribués
- Distribution des données
- Distribution des traitements
- Cloud Computing

# Conclusion





# Conclusion

- Un tour d'horizon
- Ne couvre pas toutes les possibilités
- La distribution est un des mécanismes de base utilisés pour la programmation multi-agents