Architecture Distribuée

Cours n°3

A.Saval

Objectifs du cours “Architectures distribuées”

- Compréhension des motivations
- Compréhension de la logique de conception d'une architecture distribuée
- Maîtrise des principaux modèles
- Aperçu des problèmes posés
- Aperçu de quelques frameworks existants

Aperçu du cours

- Introduction
- Problème de conception d'architecture
- Architecture logique & matérielle
- Système distribué
- Modèles d'architecture
  - Client/serveur
  - 3-tiers
  - N-tiers
  - Cluster/Cloud

MODELES D'ARCHITECTURE

Modèles existants

Client/Serveur (*2-tier*):

Collaboration entre un programme client qui envoie des requêtes et un serveur chargé d'attendre et de répondre aux requêtes.

3 niveaux (*3-tiers*):

Collaboration entre un programme client qui envoie des requêtes à un serveur d'application qui va lui-même interroger un serveur de données.

Modèles existants

N niveaux (*N-tiers*):

Généralisation du modèle à 3 niveaux, spécialisation des serveurs pour répondre à une tâche donnée.

P2P:

La séparation client/serveur disparaît; chaque nœud du modèle est capable de jouer le rôle de client ou de serveur.

Virtualisation:

Abstraction d'une ressource afin de simuler son comportement.

Historique

- Architecture “*Mainframe*”
  - Ordinateur très puissant
  - Utilisateurs connectés à des sessions virtuelles
  - Sessions distantes

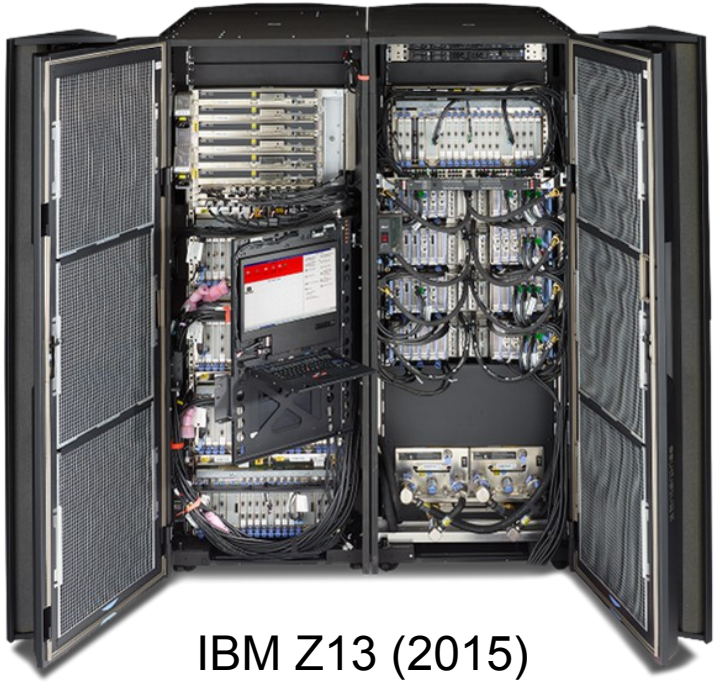
IBM 360 (1977)

Historique

- Avantage:
  - L'administration des applications est centralisée
- Inconvénient:
  - L'utilisation des resources du réseau est totalement dépendante de l'accès au *mainframe*
- Limites:
  - Les évolutions hardware/software sont complexes
  - L'apparition d'un seul problème rend le mainframe indisponible

Présent

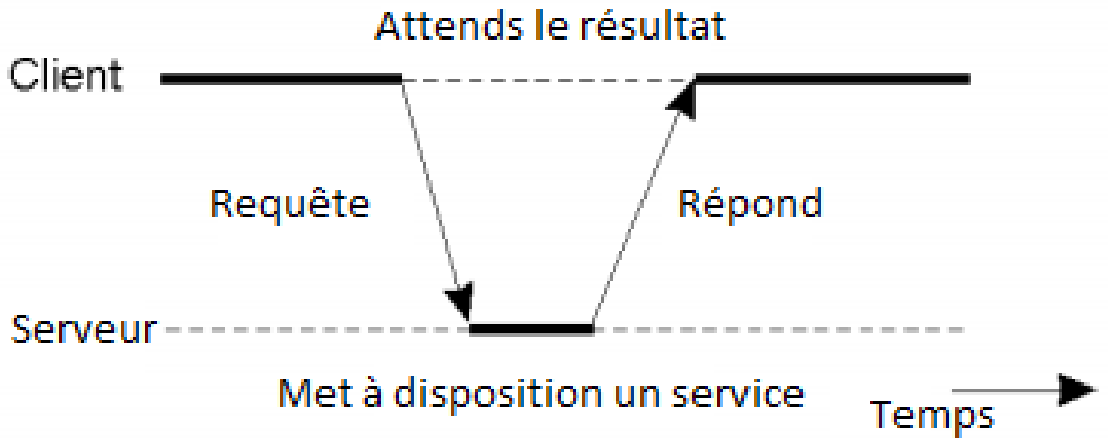
- Architecture “*Mainframe*”
  - Ordinateur très puissant
  - Utilisateurs connectés à des sessions virtuelles
  - Sessions distantes



IBM Z13 (2015)

Modèle Client/Serveur

- Permet de séparer les tâches entre
  - une application dédiée à la demande de resources ou de service: le **client.**
  - une application dédiée à la fourniture de resources ou de service: le **serveur.**



Rôle du client

- Processus:
  - établit la connexion au serveur
  - attends que la connexion soint acceptée par le serveur
  - échange des données avec le serveur en suivant un protocole de communication

- Pas de communications entre les clients

Rôle du serveur

- Processus:
  - attend une connexion entrante
  - lorqu'un client se connecte, il ouvre un socket local au système d'exploitation
  - échange des données avec le client en suivant un protocole de communication
  -
- Pas d'état destiné à représenter le client

Communications Client/Serveur

- Problèmes:
  - Comment communiquer des données à partir des applications locales pour les envoyer vers le client ?
  - Comment s'assurer que le client comprend les données échangées ?
- Solutions:
  - définir un protocole d'échanges de données.
  - définir un protocole permettant d'effectuer des appels de procédures sur un système distant.

Communications Client/Serveur

- Problèmes:
  - Comment communiquer des données à partir des applications locales pour les envoyer vers le client ?
  - Comment s'assurer que le client comprend les données échangées ?
- Solutions réalisées:
  - définir **des** protocoles d'échanges de données: TCP, UDP
  - définir **des** protocoles permettant d'effectuer des appels de procédures sur un système distant: RPC, REST, RMI, SOAP, Cobra
  - Modèle OSI

Modèle Open Systems Interconnection (OSI)

- Définition
  - Norme chargée de définir et de standardiser les fonctions nécessaires à la connexion et l'organisation dans un système communiquant

- Architecture en couches:
  - Chaque couche définit ses propres services, de protocoles et d'interfaces.

Modèle Open Systems Interconnection (OSI)

- Service:
  - description abstraite de fonctionnalités à l'aide de primitives.
- Protocole:
  - ensemble de messages et de règles d'échanges réalisant un service.
- Interface:
  - « point d'accès au service » permet d'utiliser le service.

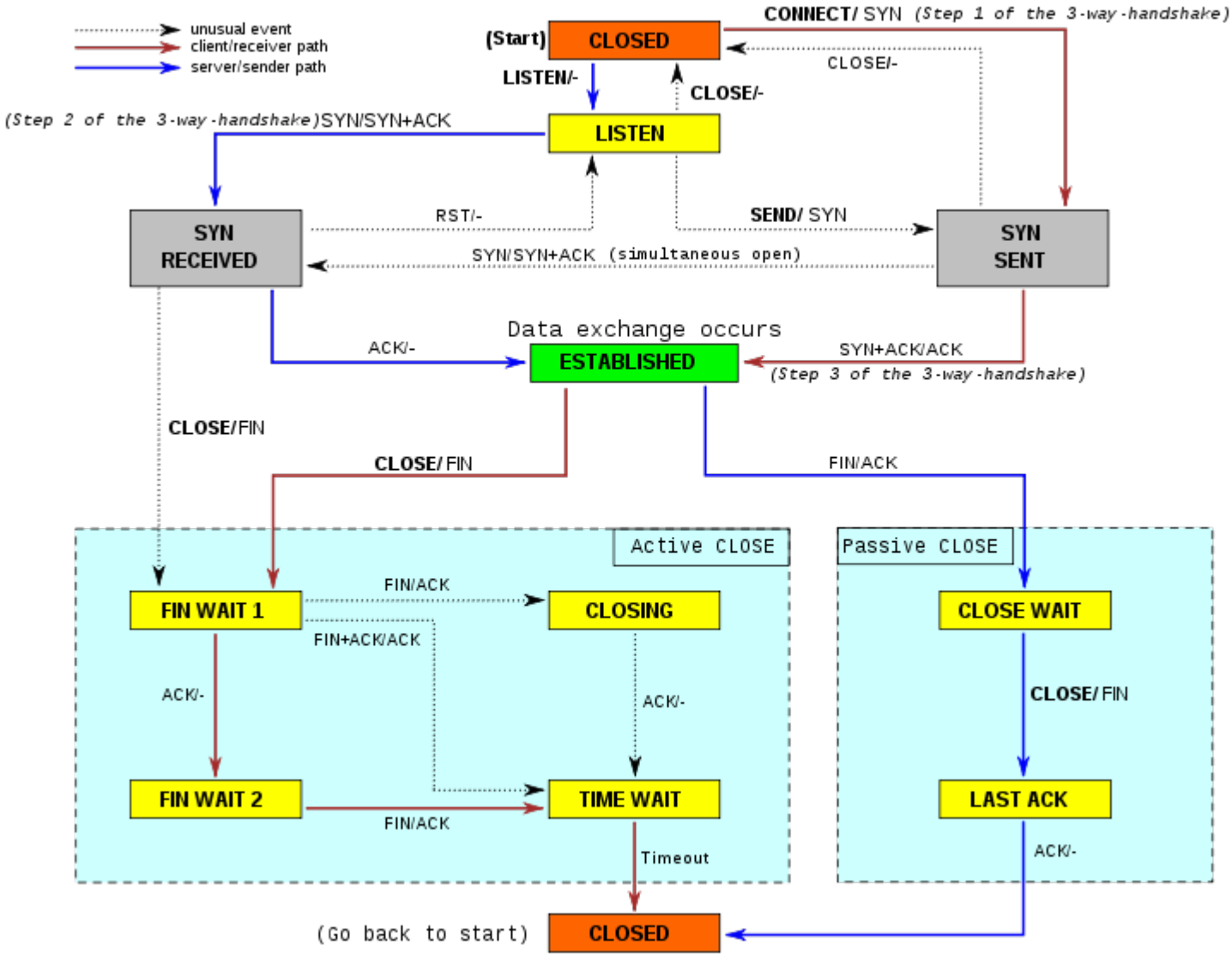
Modèle Open Systems Interconnection (OSI)

1. La couche « physique » gère la transmission des signaux entre les interlocuteurs.
2. La couche « liaison de données » gère les communications entre 2 machines reliées.
3. La couche « réseau » gère les communications de proche en proche (ex: routage et adressage des paquets)
4. La couche « transport » gère les communications entre processus.
5. La couche « session » gère la synchronisation des échanges et les « transactions ».
6. La couche « présentation » gère le codage des données applicatives.
7. La couche « application » est le point d'accès aux services réseaux. Sa définition est plus dépendante de l'application que de la norme.

Exemple: Protocole Transmission Control Protocol (TCP)

- Couche «Transport »
- Mode connecté
- Fonctionnement en trois phases :
  - établissement de la connexion
  - tranferts de données
  - Fin de la connexion

Exemple: Protocole Transmission Control Protocol (TCP)



Protocoles d'appel de procédures Serveur

- Pas de défintion stricte comme les couches OSI
- Dépendant de l'application mise à disposition
- Contraintes variables en fonction de l'environnement

  => Multiplication des modèles avec leurs avantages et leurs inconvénients

Remote Procedure Call (RPC)

- Historique: 1976 (RFC 707)
- But:
  - Permettre les appels de procédures sur un ordinateur distant à l'aide d'un serveur d'applications
- Processus:

1. Le client appelle le stub client. Cet appel est un appel de procédure locale avec des paramètres
2. Le stub client inclu les paramètres dans un message et fait au appel système pour envoyer le message.
3. Le système envoie le message à partir du poste client vers le serveur.
4. Le système sur la machine serveur transmet les paquets entrants vers le stub serveur.
5. Enfin, le stub serveur appelle la procédure serveur.

La réponse suit le même chemin étapes par étapes en sens inverse.

Representational State Transfer REST

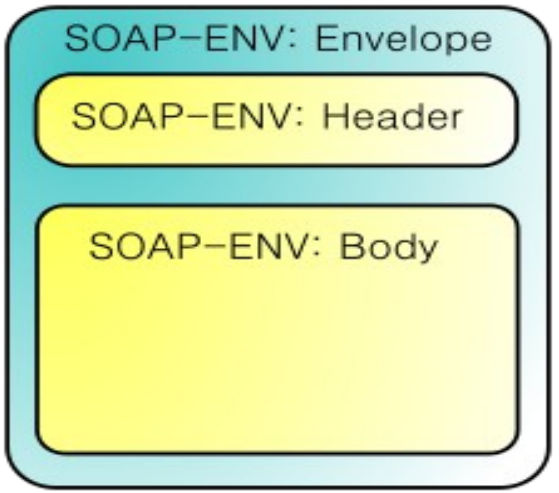
- Architecture en plus d'un protocole
- Basée sur la représentation de resources
- application au Web
  - URI nomme et identifie une ressource
  - HTTP décrit toutes les opérations; GET, POST, PUT et DELETE
  - Pas d’état
  - utilisation des standards hypermedia

Representational State Transfer REST

- Avantages
  - Simple a faire évoluer
  - Pas de gestion d'état du client sur le serveur
  - Réplication facilitée
  - Facilité d'utilisation de HTTP
  - URI pour décrire des resources
- Inconvénients
  - C'est aux clients de gérer l'état de communication avec le serveur
  - L'ensemble des appels HTTP ne sont pas supportés de manière simple par les navigateurs

Simple Object Access Protocol (SOAP)

- Protocole RPC orienté objet reposant sur XML.
- Introduit par Microsoft et IBM à l'orgine puis devenu une norme W3C,
- Le protocole SOAP est composé de deux parties :
  - une enveloppe: contenant le message et des information pour son traitement
  - un modèle de données: qui décrit le format du message



Simple Object Access Protocol (SOAP)

- Avantages
  - Compatible avec d'autres protocoles de transports
  - Indépendant de la plateforme
  - Indépendant du langage
  - Extensible
- Inconvénients
  - Augmente la taille de paquets de données à transporter
  - Crée une dépendance dans la façon doivent communiquer le serveur et le client

MODELES D'ARCHITECTURE
modèle *3-tiers*

Références

- Software Architecture: IEEE Standard 1471-2000
- P. Kruchten, Architectural Blueprints—The “4+1” View Model of Software Architecture, IEEE Software 12 (6), Nov. 1995, pp42-50
- Tanenbaum & van Steen, Distributed Systems, Principles and Paradigms, seconde édition
- Architecture of Distributed Systems, cours de Johan Lukkien, 2011
- Architectural Patterns Revisited – A Pattern Language, Paris Avgeriou & Uwe Zdun, 2005
- Software Architecture, Foundations, Theory, and Practice, R.N. Taylor, N. Medvidovic, E.M. Dashofy, Wiley & Sons, 2009
- Software Architecture in Practice, Second Edition, L. Bass, P. Clements, R. Kazman, SEI Series in Software Engineering, Addison-Wesley, 2003