Introduction aux applications réparties

Noël De Palma
Projet SARDES
INRIA Rhône-Alpes
http://sardes.inrialpes.fr/~depalma
Noel.depalma@inrialpes.fr

Applications réparties

Def : Application s'exécutant sur plusieurs sites reliés par un réseau de communication

- Intérêt des applications réparties
- Quelques exemples
- Caractéristiques et besoins
- Modèle de communication
- Exemple

Evolution des insfrastructures

- Cluster où grappe
  - Interconnexion de serveurs sur un LAN
  - 1 domaine d'administration
- Grid où grille
  - Interconnexion de grappe sur un LAN/WAN
  - N domaine d'administration
- Peer-to-Peer où P2P
  - Structuré ou non-structuré
  - Pas de domaine d'administration

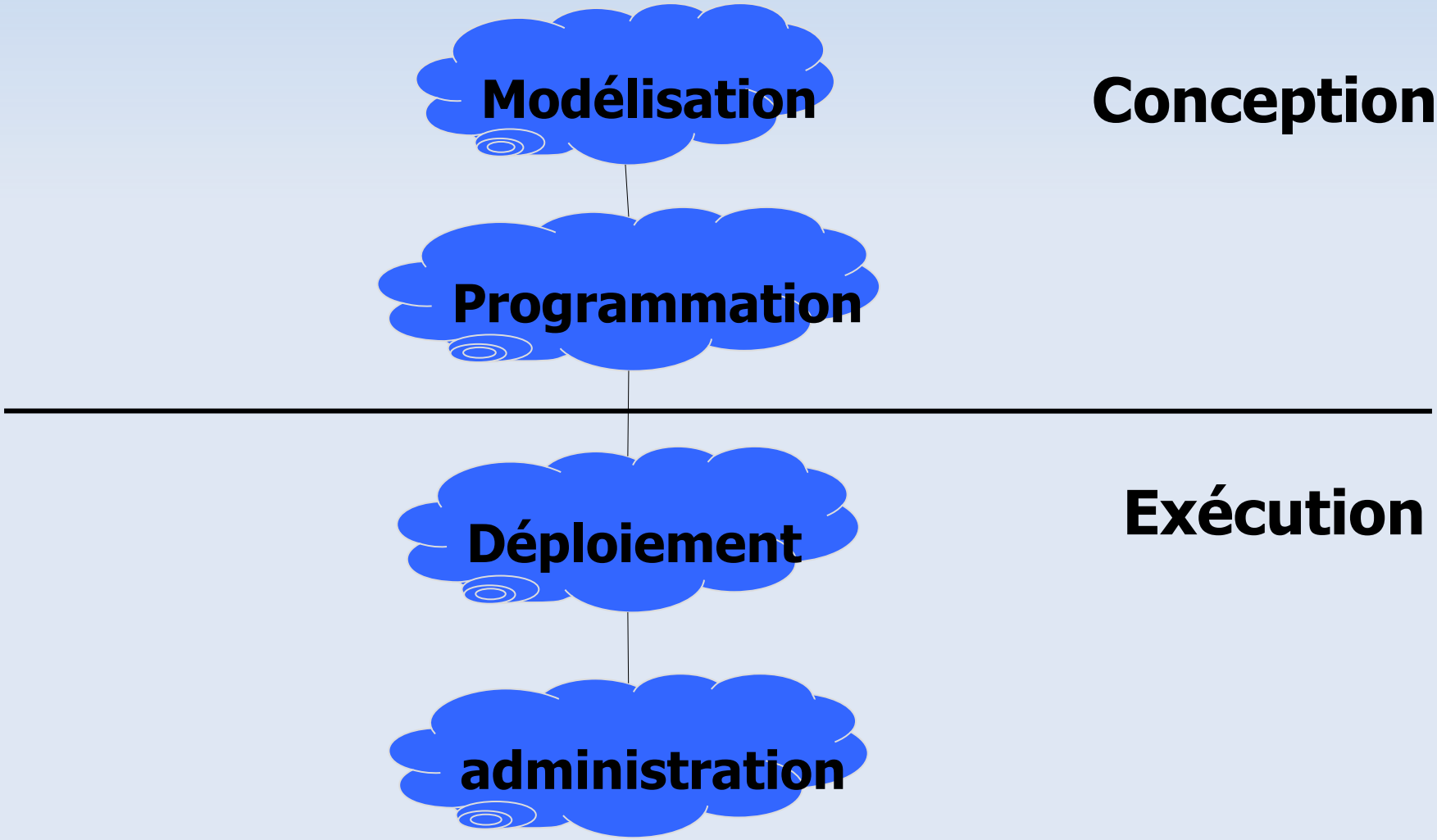
Evolution des infrastructures

- Cloud computing
  - Fournir des ressources à la demande
    - Machine et réseau virtuelle
    - Auto scalabilité (scale up/scale down)
    - Pay as you use : faible coût de déploiement
  - 3 couches : IaaS/PaaS/SaaS
  - 3 modèles
    - Public, private, hybrid
  - Ex : Amazon EC2, Google Apps, MS AZURE

Intérêt des applications réparties

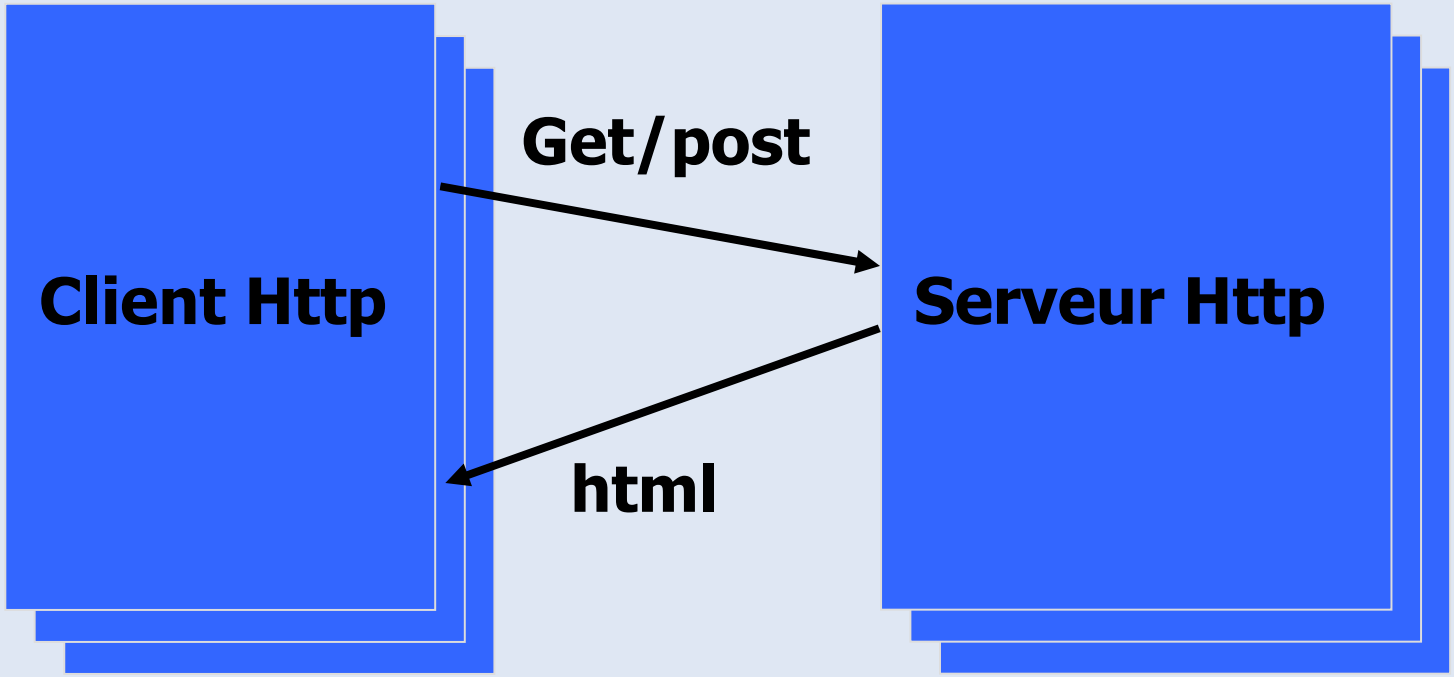
- Besoin de communication et de partage d'information
- Besoin de puissance de calcul
- Besoin de tolérance aux fautes

Cycle de vie


Modélisation
Programmation
Déploiement
administration
Conception
Exécution

Web

- Accès universel à de l ’information

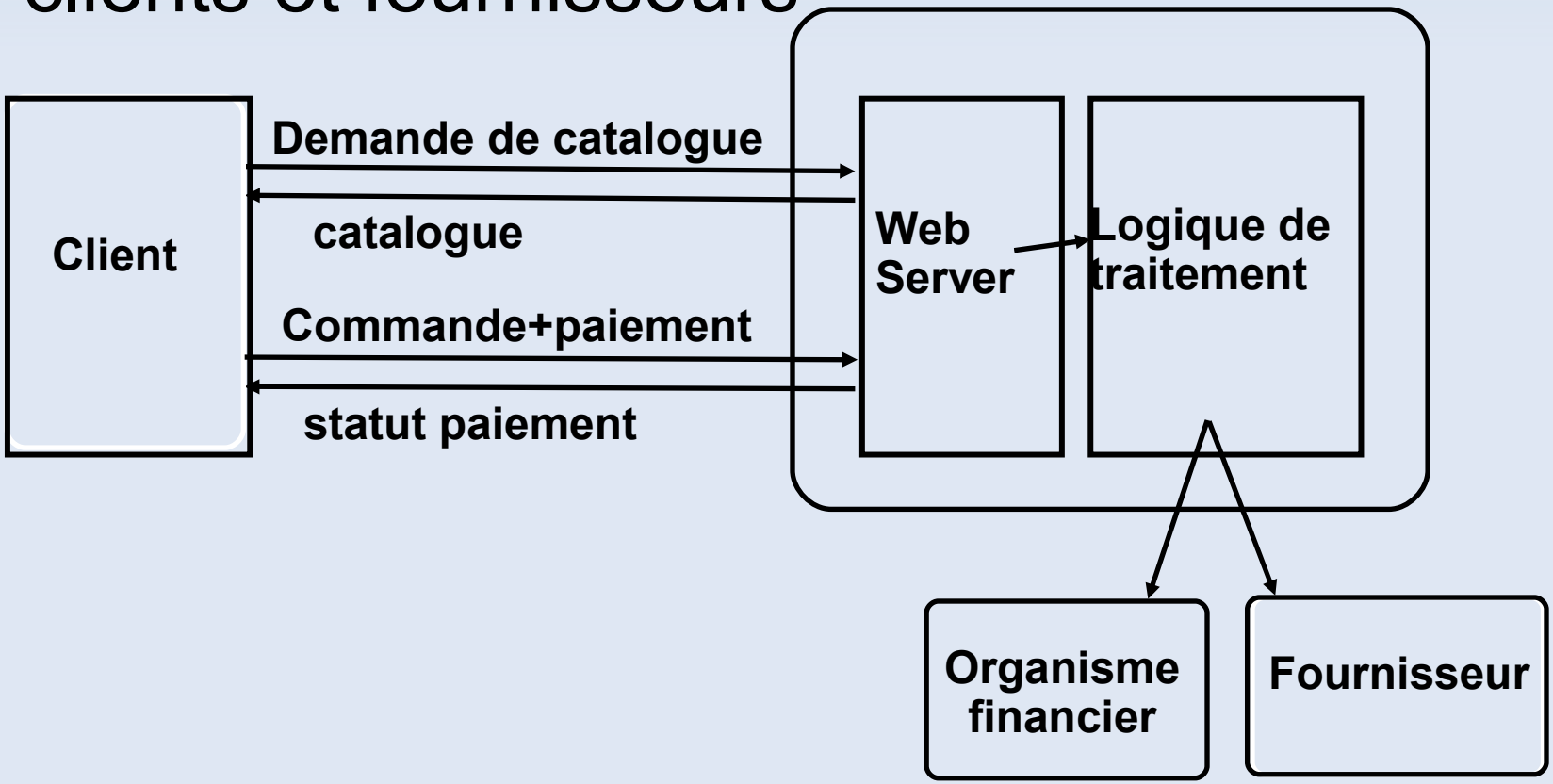


Problèmes de base

- Liaison
- Hétérogénéité

Commerce électronique

- Transactions commerciales entres clients et fournisseurs



Problèmes de base

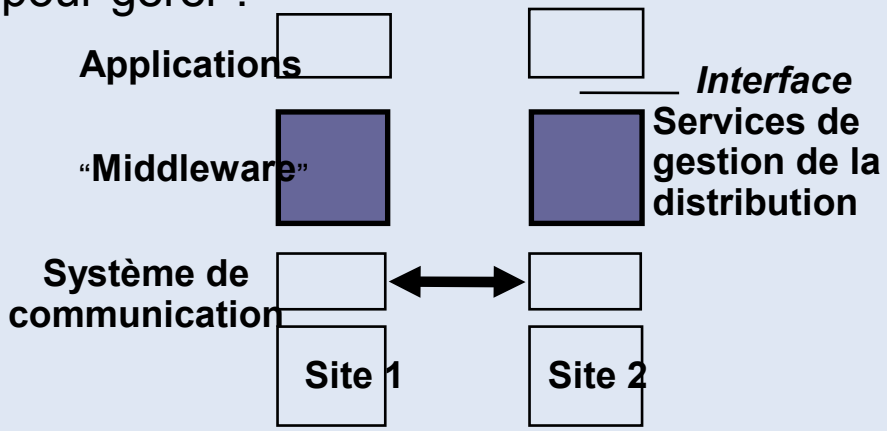
- Liaison, Hétérogénéité
- Passage à l'échelle (Scalabilité)
- Sécurité
- Fiabilité
- Qualité de service
- Intégration (fournisseur, organisme financier)

Caractéristiques des applications réparties

- Distribution des ‘ composants ’ de l’application
  - Liaison
  - hétérogène
  - Fiabilité
  - Sécurité
  - Scalabilité
  - Persistance
  - Intégration
  - Performance
  - Administration
- ***=> Plus difficiles à concevoir, à programmer, à déployer et à administrer => besoin de modèles & d’outils adaptés***

Outils de programmation

- Outils de base : socket
- Middleware (Intergiciel)
  - Couche logiciel (répartie) destinée à :
    - Faciliter la programmation réseau
    - Masquer la répartition des traitements et des données
    - Masquer l ’hétérogénéité des machines & des systèmes
    - Faciliter l’interopérabilité des applications
    - Fournir des services associés pour gérer :
      - Fiabilité
      - Sécurité
      - Scalabilité
      - …



Modèles de structuration d'application répartie

- Différents Modèle de communication
  - Modèle client serveur
  - Modèle à messages
  - Modèle à base de code mobile
  - Modèle à mémoire partagée
- Différentes infrastructures
  - Grappes, grilles, p2p
- Différents types d ’intergiciels pour adresser différents problèmes
- Analyse du problème => choix de l’ intergiciel

Quelques Modèles de structuration d'application répartie

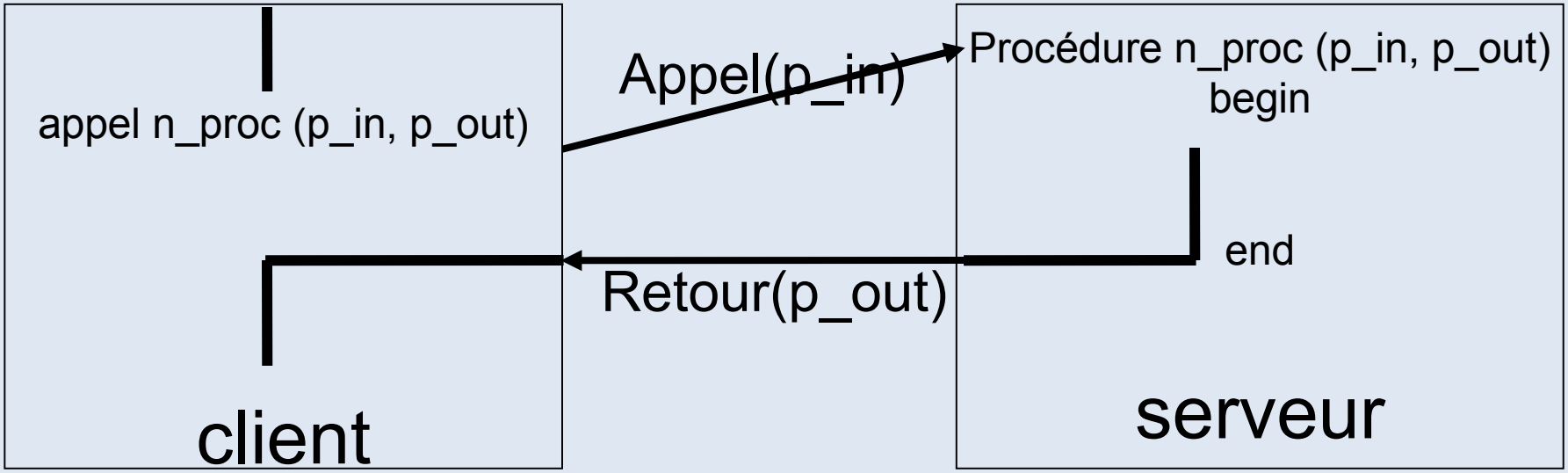
- Client/serveur
- Modèle à messages
- Modèle à base de code mobile
- Modèle à mémoire partagée

Modèle client-serveur définition

- Application client/serveur
  - application qui fait appel à des services distants au travers d'un échange de messages (les requêtes) plutôt que par un partage de données (mémoire ou fichiers)
  - serveur
    - programme offrant un service sur un réseau (par extension, machine offrant un service)
  - client
    - programme qui émet des requêtes (ou demandes de service). Il est toujours l'initiateur du dialogue

Modèle client-serveur
Interactions

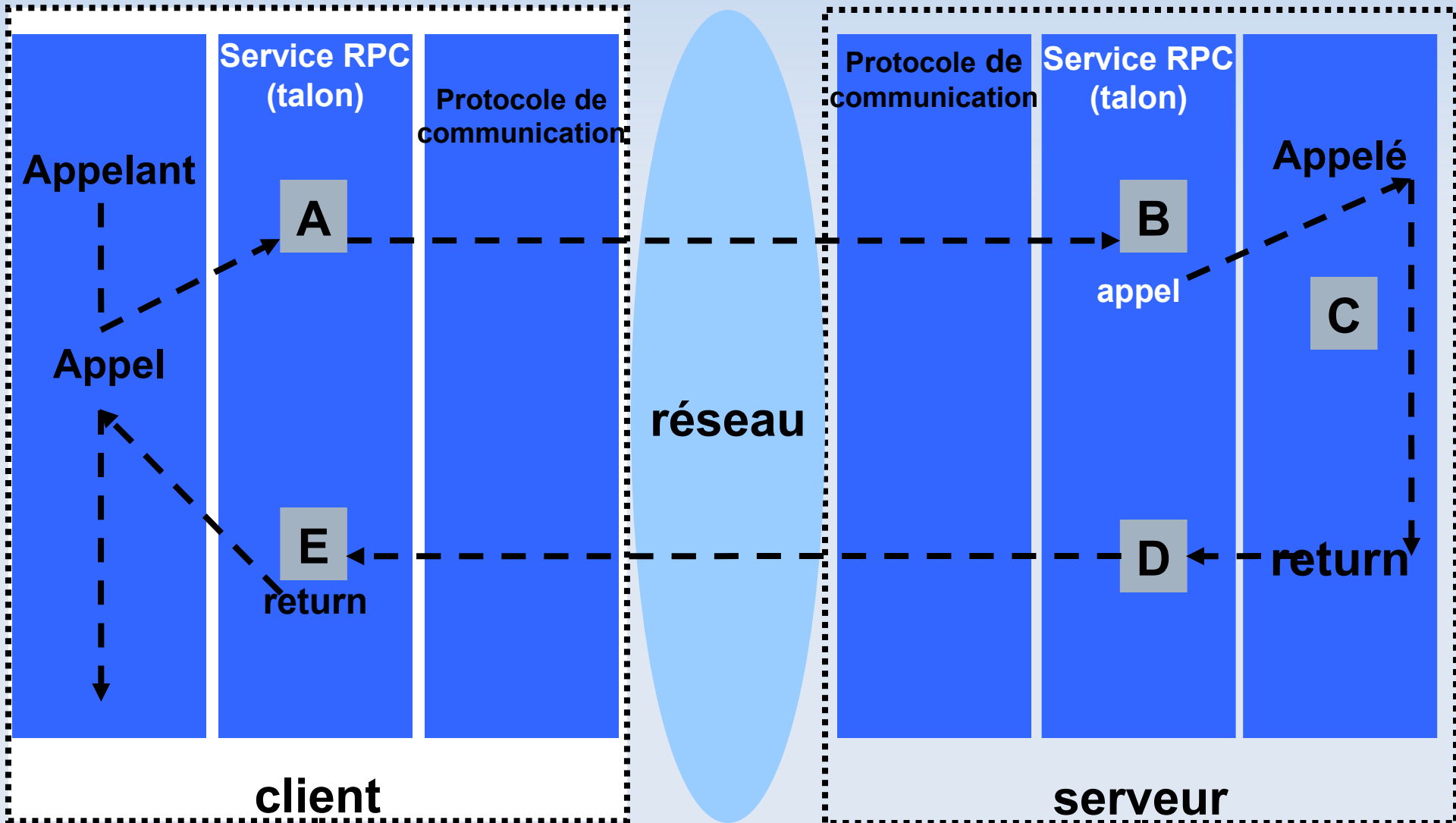
- Deux messages (au moins) échangés
  - Le premier message correspondant à la requête est celui de l'appel de procédure, porteur des paramètres d'appel.
  - Le second message correspondant à la réponse est celui du retour de procédure porteur des paramètres résultats.



Appel de procédure à distance (RPC)

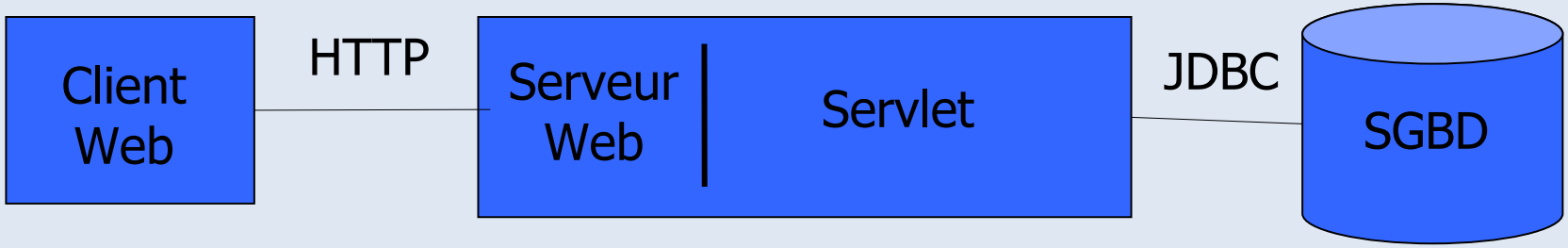
- Outils de base pour réaliser le mode client-serveur.
  - L'opération à réaliser est présentée sous la forme d'une procédure que le client peut faire exécuter à distance par un autre site : le serveur.
    - Forme et effet identique à ceux d'un appel local
      - Simplicité (en l'absence de pannes)
      - Sémantique identique à celle de l'appel local

RPC [Birrel & Nelson 84]
Principe de réalisation

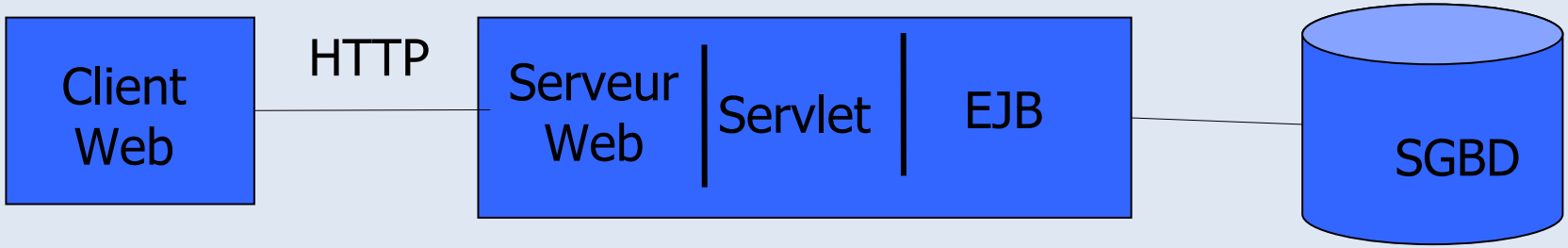


Evolution des architectures client/serveur : application 3-tiers

- Présentation/logique applicative/données
- Implantation ad-hoc



- J2EE

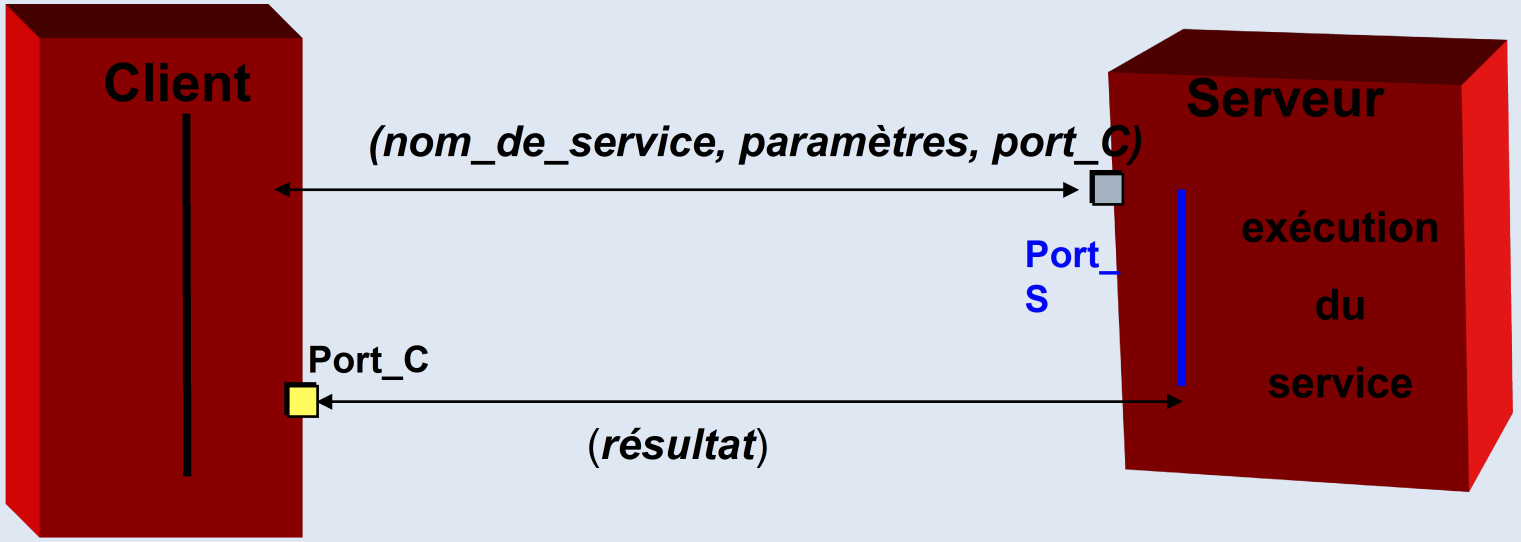


Modèle de communication par messages

- Message Passing (communication par messages)
- Message Queuing (communication par file de message)
- Publish/Subscribe (communication par abonnements)

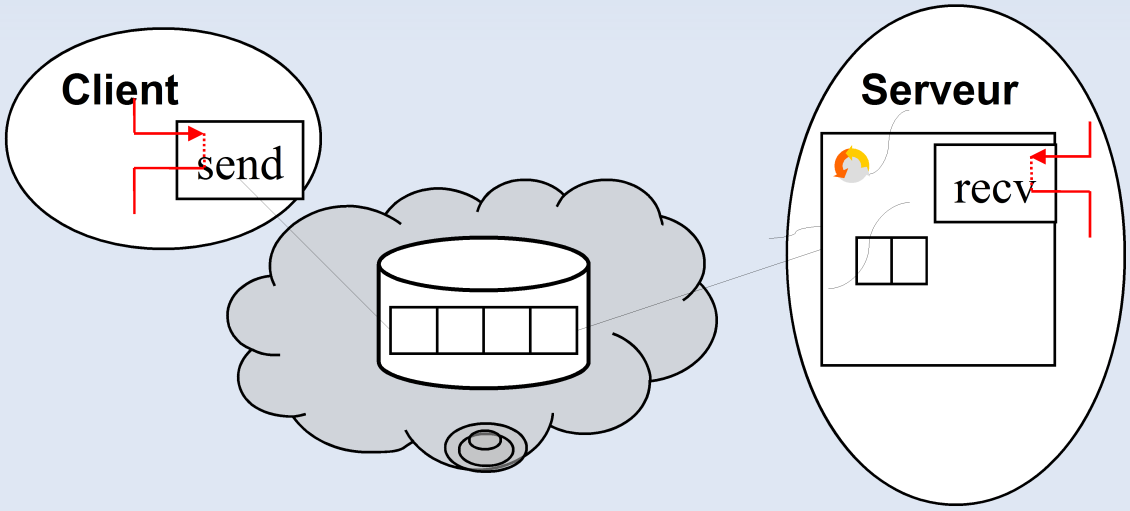
Message passing

- Principes directeurs
  - communication asynchrone
  - communication directe ou indirecte (via des “portes”)
    - problème de désignation, localisation des entités coopérantes
  - messages éventuellement typés



Message Queuing

- Queue de messages
  - persistantes ➜ asynchronisme et fiabilité

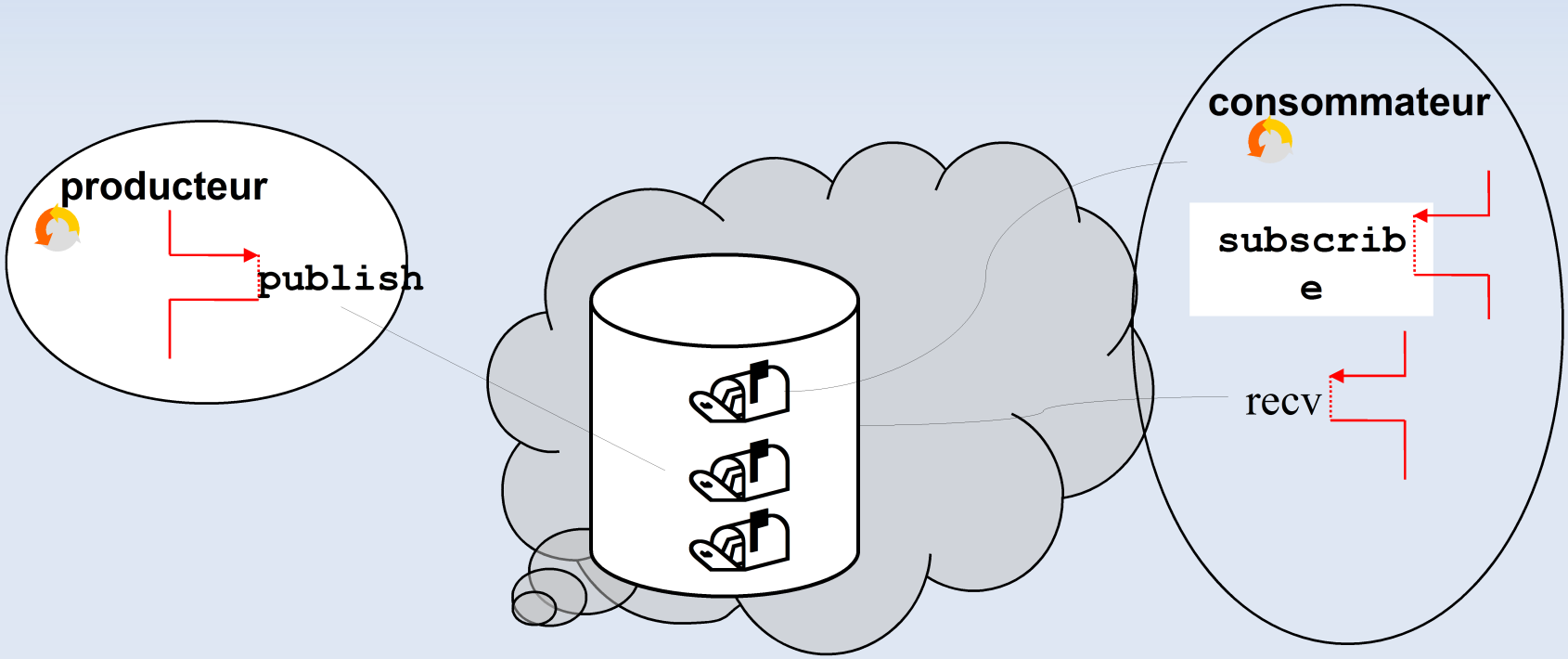


- Indépendance de l'émetteur et du destinataire
  - Le destinataire n’est pas forcément actif

Publish/Subscribe (1)

- Désignation anonyme
  - L'émetteur envoie un message
    - Basé sur un sujet (subject-based)
    - Basé sur un contenu (content-based)
  - Le récepteur s'abonne (à un sujet ou un contenu)
- Communication 1-N
  - Plusieurs récepteurs peuvent s'abonner

Publish/Subscribe (2)



Code mobile

- Définition : programmes pouvant se déplacer d'un site à un autre
  - Function shipping versus data shipping
- Motivations
  - Rapprocher le traitement des données
  - Réduire le volume de données échangées sur le réseau
- Caractéristiques
  - Code interprétable
  - Problèmes particuliers de la sécurité
  - Exemples : web/javascript

Modèle à mémoire partagé

- Principe
  - Simulation d'une mémoire globale partagée
- Exemple : Modèle à objets répartis partagés
  - Espace d'objets réparties partagés
  - Réalisé par des objets répliqués
  - Problèmes de cohérence

Un exemple d ’application

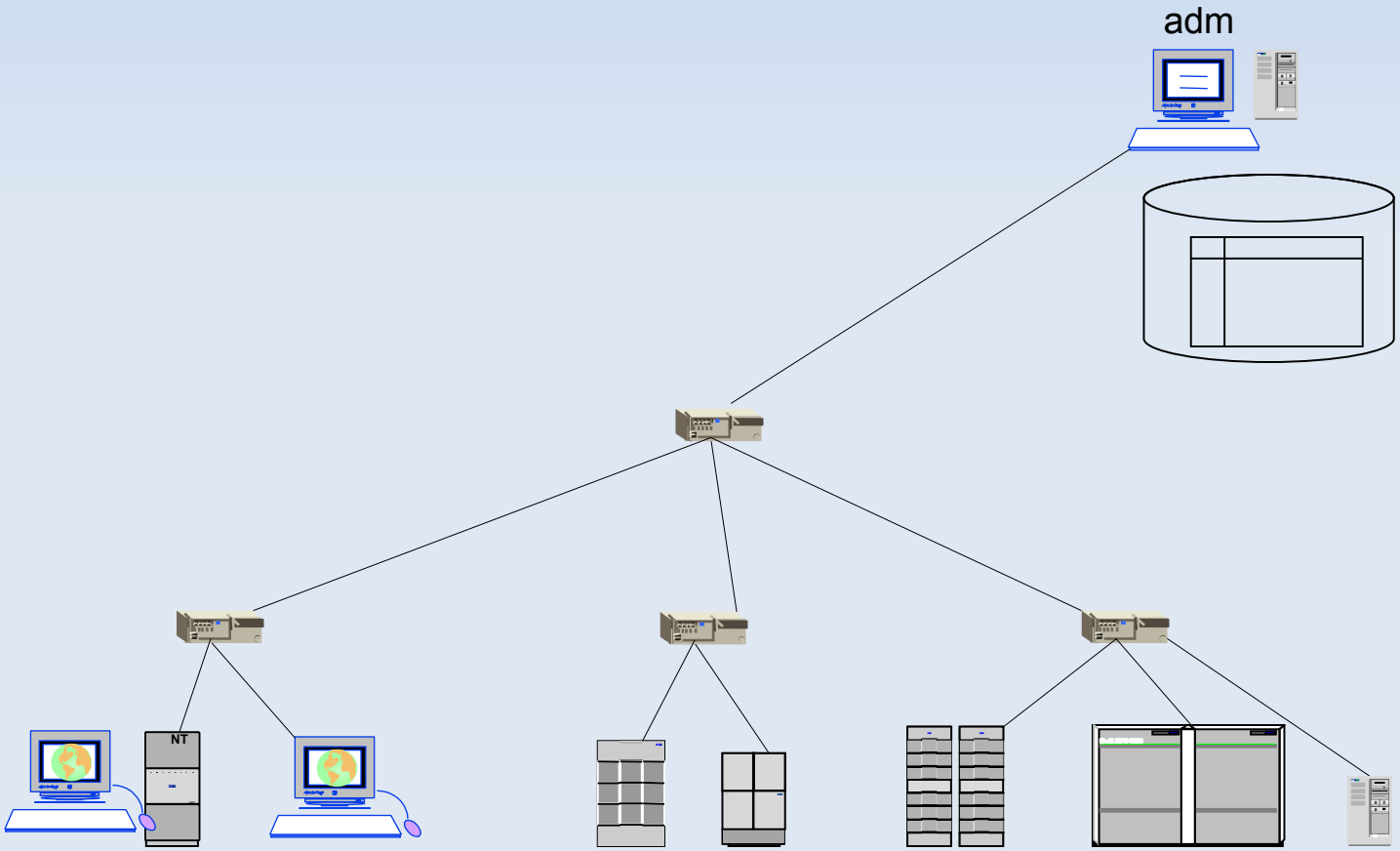
Surveillance des équipements d ’un réseau

- Problème :
  - Surveillance de l'état de machines, de systèmes d'exploitation et d'applications dans un environnement distribué.
  - Flot continuel de données en provenance de sources diverses sur le réseau.
  - Les éléments du système peuvent apparaitre, disparaitre, migrer, etc.

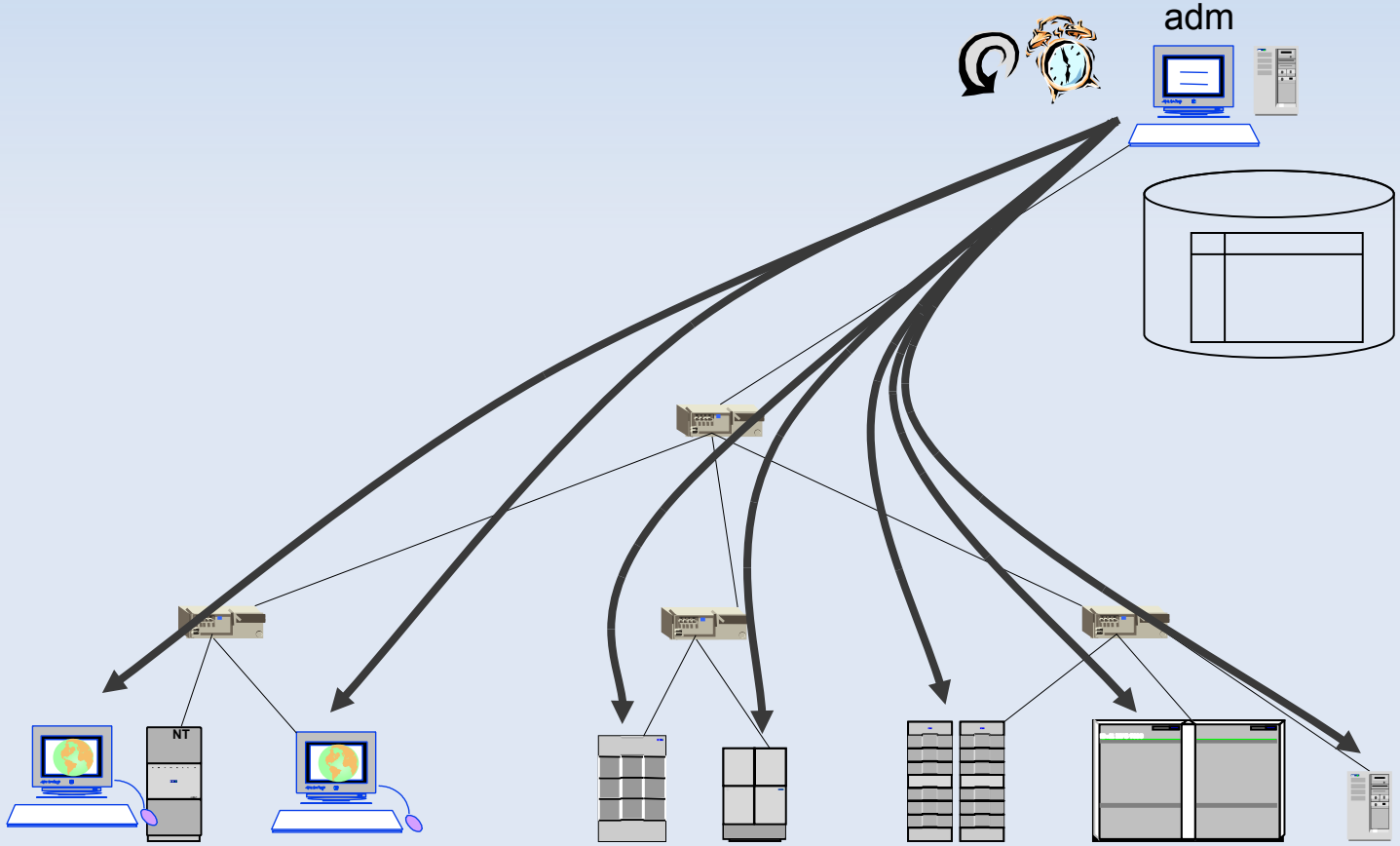
Solution client/serveur

- Interrogation régulière
  - par l'application d'administration (client)
  - des éléments à surveiller (serveur)
  - mise à jour d'une base de données centralisée.
    - Utilisation d'une configuration complexe afin de connaitre l'ensemble des éléments à surveiller.
    - Maintien de cette configuration lorsque des machines ou des applications rejoignent, quittent ou se déplacent dans le système.
  - Interrogation par les administrateurs de la base centrale.

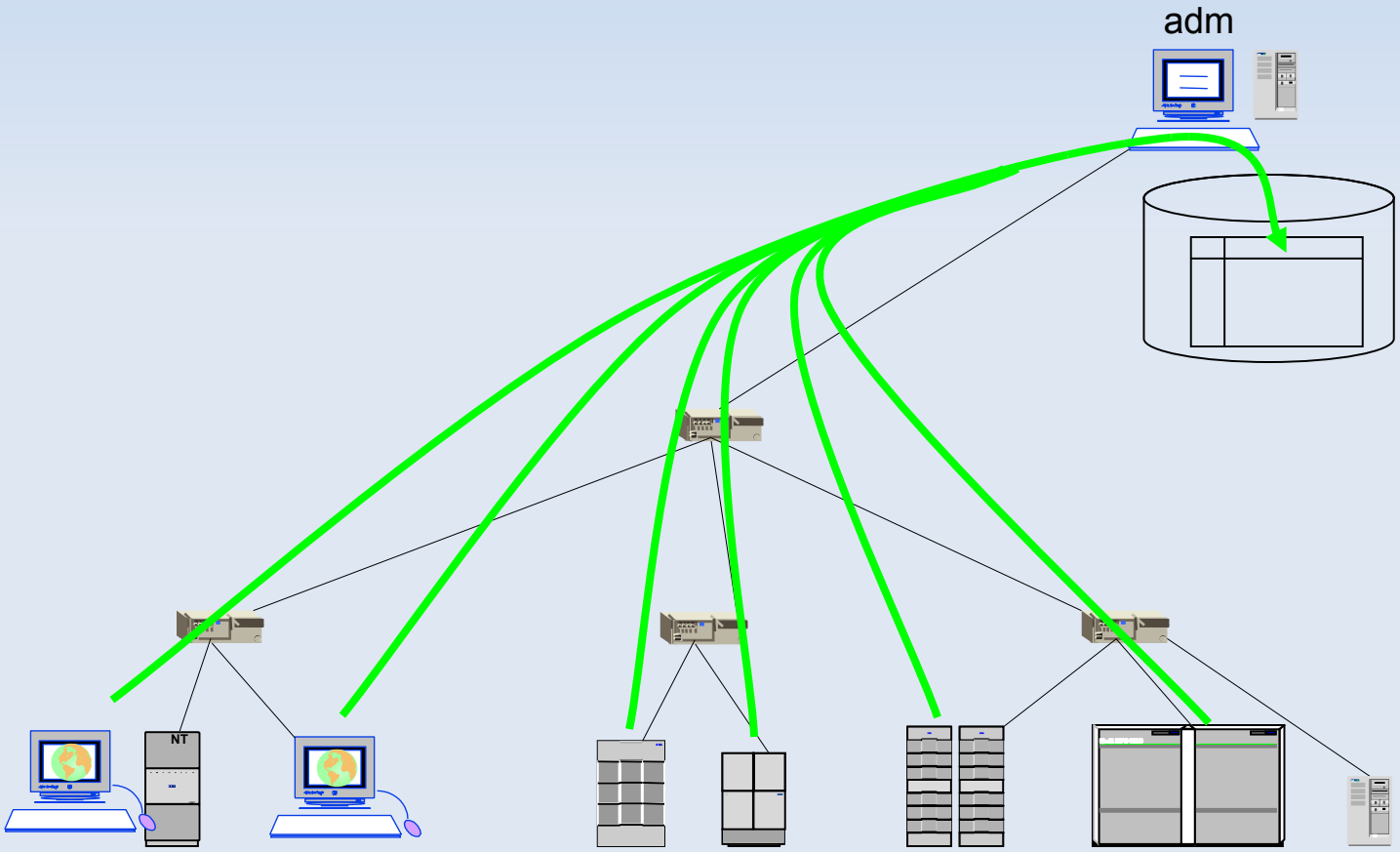
Exemple d ’application
Solution client-serveur



Exemple d ’application
Solution client-serveur (pull)



Exemple d ’application
Solution client-serveur

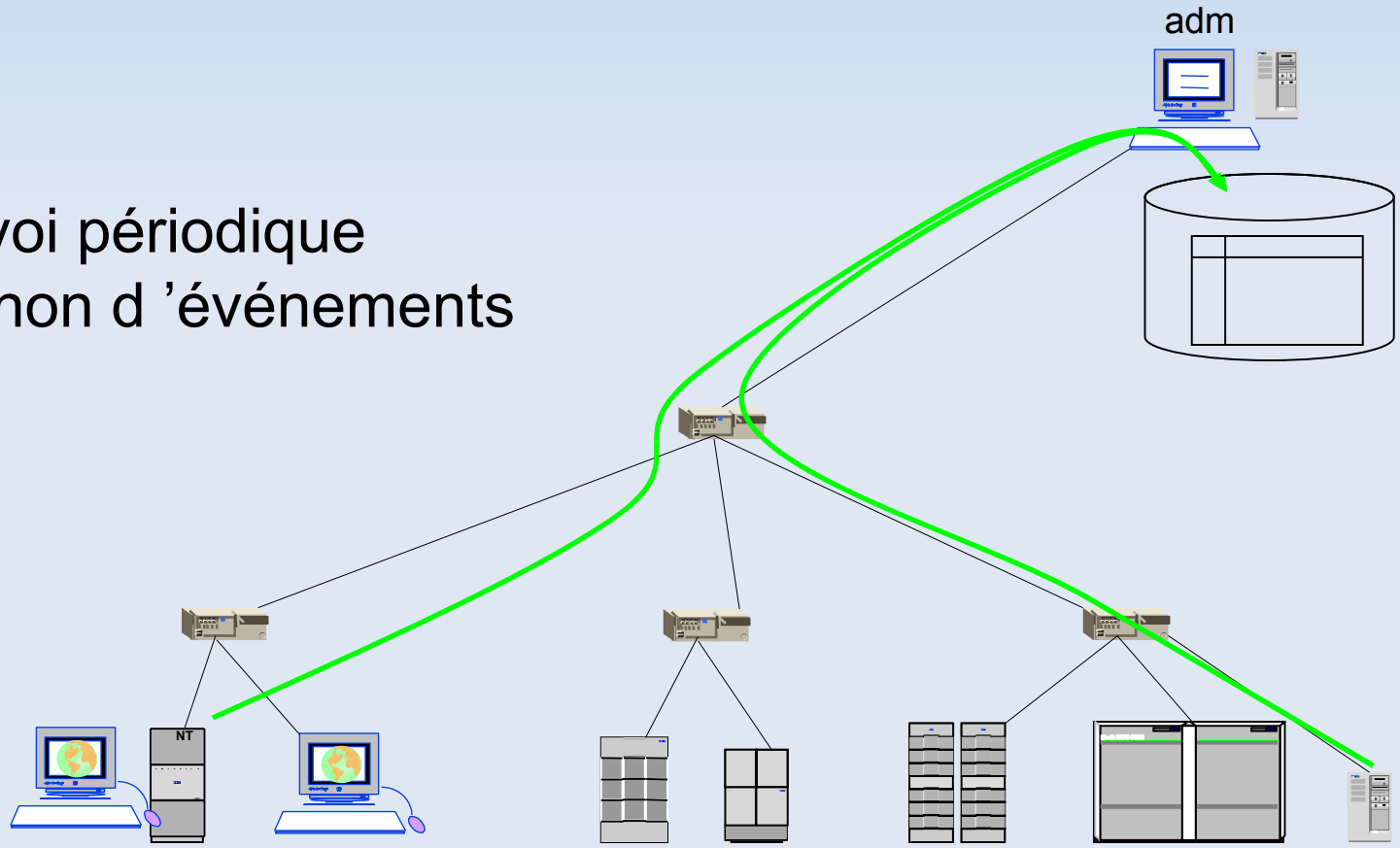


Exemple d ’application
Solution « bus à messages »

- Solution "Messaging" :
  - Les différents éléments administrés émettent des messages :
    - changements d'état et de configuration
    - alertes, statistiques
  - Un ou plusieurs démons reçoivent ces notifications et maintiennent l'état courant du système
    - suivi des changements de configuration dynamiques

Exemple d ’application
Solution bus à messages (push)

Envoi périodique
ou non d ’événements



Exemple d ’application
Solution bus à messages

Lecture de la base d ’information au moment souhaité