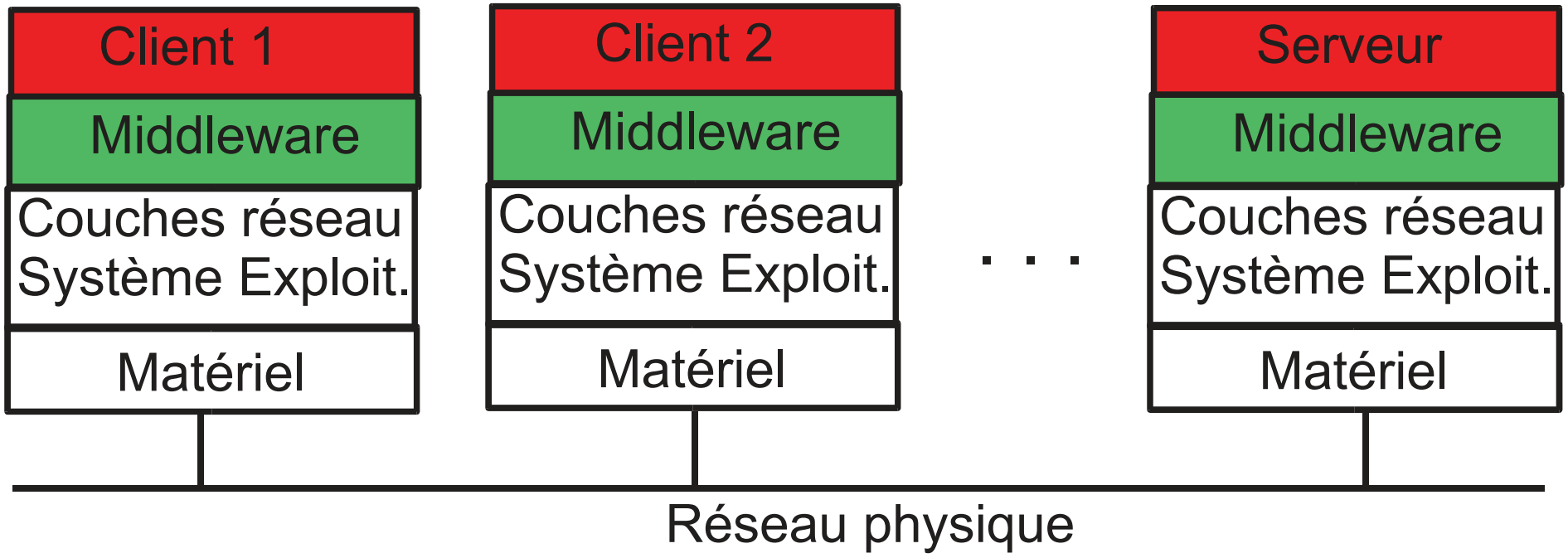
Middleware

- Middleware (intergiciel en français) : couche logiciel
  - S'intercale entre le système d'exploitation/réseau et les éléments de l'application distribuée
  - Offre un ou plusieurs services de communication entre les éléments formant l'application ou le système distribué
    - Services de plus haut niveau que les communications via sockets TCP/UDP, augmentation du niveau d'abstraction



Middleware

- Plusieurs grandes familles de middleware
  - Appel de procédure/méthode à distance
    - Extension « naturelle » de l'appel local d'opération dans le contexte des systèmes distribués
      - Une partie serveur offre une opération appelée par une partie client
    - Permet d'appeler une procédure/méthode sur un élément/objet distant (presque) aussi facilement que localement
    - Exemples (dans ordre historique d'apparition)
      - RPC (Remote Procedure Call) : solution de Sun pour C/Unix
      - CORBA (Common Object Request Broker Architecture) : standard de l'OMG permettant l'interopérabilité quelque soit le langage ou le système
      - Java RMI (Remote Method Invocation) : solution native de Java
  - Envoi ou diffusion de messages ou événements
    - Famille des MOM (Message Oriented Middleware)
    - Event Service de CORBA, JMS de Java
  - Mémoire partagée
    - Accès à une mémoire commune distribuée
    - JavaSpace de Java

Middleware

- Familles de middlewares, caractéristiques générales
  - Modèle RPC/RMI
    - Interaction forte entre parties client et serveur
      - Appel synchrone (pour le client) d'opération sur le serveur
      - Généralement, le client doit explicitement connaître le serveur
      - Peu dynamique (point de vue serveur) car les éléments serveurs doivent être connus par les clients et lancés avant eux
    - Mode 1 vers 1 : opération appelée sur un seul serveur à la fois
  - Modèles à message ou mémoire partagée
    - Asynchrone et pas d'interaction forte entre éléments
      - Envoi de message est asynchrone
      - Pas de nécessité de connaître les éléments accédant à la mémoire
      - Permet une plus forte dynamicité : ajout et disparation d'éléments connectés au middleware facilités par les faibles interactions et l'anonymat
    - Mode 1 vers n
      - Diffusion de messages à plusieurs éléments en même temps
      - Accès aux informations de la mémoire par plusieurs éléments

Middleware

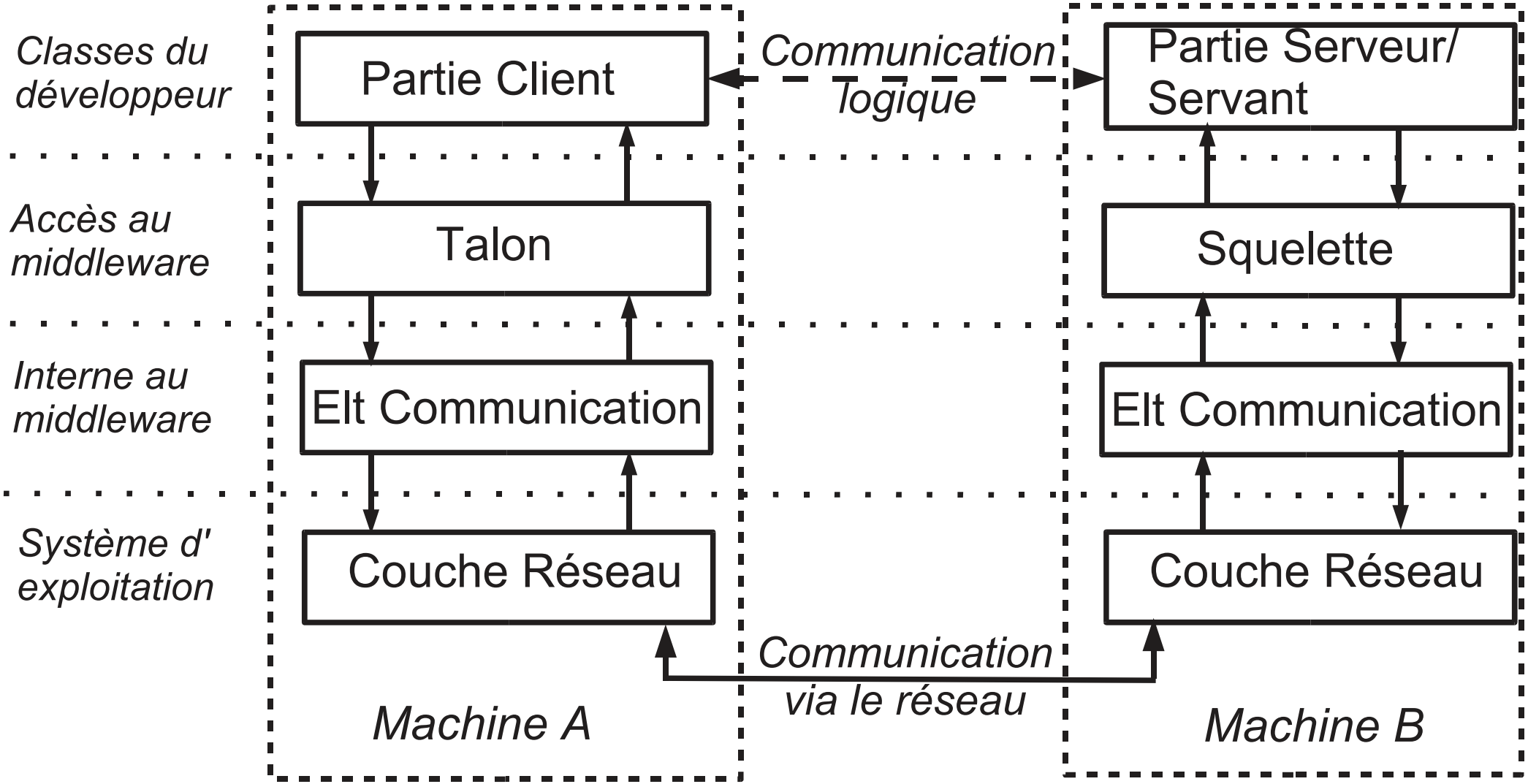
- Caractéristiques que peut assurer un middleware
  - Transparence à la localisation
    - Accès à des objets/ressources distantes aussi facilement que localement et indépendamment de leur localisation
      - Cas pour tous les middleware
  - Support de l'hétérogénité des systèmes et des langages
    - CORBA par exemple : fait interopérer via du RPC/RMI des éléments logiciels écrits dans n'importe quel langage
  - Tout autre type de transparence (voir cours intro), selon le middleware
- Services offerts par un middleware
  - Un ou plusieurs services de communication/interaction
    - RPC, diffusion de message, mémoire partagée ...
  - Service de nommage

Middleware

- Service de nommage du middleware
  - Pour enregistrer, identifier et rechercher les éléments et services connectés via le middleware
- Service de nommage, cas des middlewares RPC/RMI
  - Service d'un élément = (interfaces d') opération(s) offerte(s) par un élément et pouvant être appelée(s) par d'autres
  - Un élément enregistre les opérations qu'il offre
  - Recherche d'une opération ou d'un élément, 2 modes
    - On demande à recevoir une référence sur un élément qui offre une opération compatible avec celle que l'on cherche
    - On sait identifier l'élément distant et on demande à récupérer la référence sur cet élément
  - Une fois la référence sur l'élément distant connu, on peut appeler l'opération sur cet élément via cette référence

Middleware RPC/RMI

- Architecture générale, plusieurs couches



Middleware RPC/RMI

- Éléments de l'architecture générale
  - 2 catégories
    - Éléments communiquants réalisés par le développeur
    - Éléments permettant la communication
- Description des différents éléments
  - Éléments dont le code est généré automatiquement via des outils du middleware
    - Talon (stub) : élément/proxy du coté client qui offre localement les mêmes opérations (la même interface) que le servant
    - Squelette (skeleton) : élément du coté serveur qui reçoit les requêtes d'appels d'opérations des clients et les lance sur le servant

Middleware RPC/RMI

- Description des différents éléments (suite)
  - Partie client : partie qui appelle l'opération distante
  - Partie serveur : partie qui offre l'opération distante via une interface
    - Appelée aussi « servant » pour certains middleware
    - A implémenter par le développeur : contient le code des opérations de l'interface
  - Des 2 cotés : éléments de communication entre parties client/serveur
    - Internes au middleware
    - Attaquent les couches TCP ou UDP via des sockets pour gérer la communication entre les talons et squelettes

Middleware RPC/RMI

- Fonctionnement général d'un appel d'opération à distance

1. La partie client récupère via le service de nommage une référence d'objet sur le servant distant
   - En réalité, une référence sur le talon local
2. La partie client appelle une opération sur cette référence d'objet
   - Appel synchrone : la partie client attend que l'appel de l'opération soit terminé pour continuer son exécution
3. Le talon, qui reçoit cet appel, « compacte » (marshall) toutes les données relatives à l'appel (identificateur opération + paramètres)
4. L'élément de communication envoie les données à l'élément de communication coté serveur
5. L'élément de communication coté serveur envoie les données au squelette
6. Le squelette décompacte les données (unmarshall) pour déterminer l'opération à appeler sur le servant
7. L'opération est appelée sur le servant par le squelette

Middleware RPC/RMI

- Fonctionnement général d'un appel d'opération à distance (fin)

8. Le squelette récupère la valeur de retour de l'opération, la compacte et l'envoie au talon via les éléments de communication
   - Même si pas de valeur de retour, on renvoie quelque chose au talon pour l'informer de la fin de l'appel d'opération
9. Le talon décompacte la valeur et la retourne la valeur à la partie client
10. L'appel de l'opération distante est terminé, la partie client continue son exécution

Middleware RPC/RMI

- Gestion des problèmes de communication entre partie client et servant
  - Problèmes potentiels
    - Le servant est planté ou inaccessible
    - La requête d'appel d'opération n'a pas été reçue coté servant
    - La réponse à la requête n'a pas été reçue par la partie client
    - L'exécution de l'opération s'est mal passée
    - ...
  - En cas de problèmes, coté client, on n'a pas reçu la réponse à la requête
    - Doit renvoyer la requête au serveur a priori
    - Mais coté serveur, si l'opération modifie un état doit éviter de rappeller l'opération a priori
    - Plusieurs cas possibles pour gérer cela

Middleware RPC/RMI

- 4 sémantiques possibles pour l'appel d'opérations
  - « Peut-être »
    - On ne sait pas si l'opération a été appelée
  - « Au moins une fois »
    - L'opération a été appelée au moins une fois mais peut-être plusieurs fois également
      - Problème si opération modifie un état (opération non idempotente)
  - « Au plus une fois »
    - L'opération est appelée une seule fois ou on reçoit un message informant qu'un problème a eu lieu et que l'opération n'a pas été appelée
  - « Une fois »
    - Cas idéal mais difficile à atteindre

Middleware RPC/RMI

- Trois caractéristiques pour gérer les requêtes d'appels d'opérations
  - Retransmission de la requête
    - Le client peut redemander au serveur d'appeler l'opération en cas de non réponse reçue de sa part
  - Filtrage des duplicats de requêtes
    - Si le serveur détecte qu'il s'agit d'une redemande d'appel d'opération, il n'exécute pas une nouvelle fois l'opération
  - Retransmission des résultats
    - Le serveur garde un historique des valeurs de retour des opérations et peut renvoyer le résultat retourné pour une opération en cas de nouvelle demande de cette même opération

Middleware RPC/RMI

- Trois combinaisons principales de caractéristiques et sémantiques associées
  - Pas de retransmission de la requête
    - Sémantique « peut-être »
  - Retransmission des requêtes mais pas de filtrage des duplicats de requêtes
    - Sémantique « au moins une fois »
  - Retransmission des requêtes, filtrage des duplicats de requêtes et utilisation de l'historique des réponses
    - Sémantique « au plus une fois »

Garbage collector distribué

- Dans la plupart des mises en oeuvre de langages objets : garbage collector
  - « Ramasse miette » en français
  - But : supprimer de la mémoire locale (du programme, de la JVM en Java) les objets n'étant plus référencés par aucun autre objet
- Objets distribués (via un middleware comme Java RMI)
  - Capacité à référencer des objets distants comme des objets locaux
  - Pour nettoyage des objets non référencés : doit alors prendre aussi en compte les références distantes
  - Distributed Garbage Collector (DGC)

Garbage collector distribué

- Fonctionnement général d'un garbage collector (en local)
  - Un compteur de référence est associé à chaque objet
  - Quand un objet récupère une référence sur l'objet, on incrémente de 1 le compteur de cet objet
  - Quand un objet ne référence plus l'objet, on décrémente le compteur de 1
  - Quand compteur a pour valeur 0, on peut supprimer l'objet de la mémoire car plus aucun objet ne l'utilise plus
- En distribué peut utiliser un fonctionnement similaire
  - En prenant en compte à la fois les références locales et distantes

Garbage collector distribué

- Contraintes supplémentaires en distribué
  - Coordination entre garbages collectors sur chaque machine pour assurer gestion des références
    - Référence sur un objet distant = référence locale sur un proxy pour cet objet
    - Quand localement un proxy n'est plus référencé par aucun objet, on peut informer le GC gérant l'objet distant de la perte de référence
  - Si une machine distante est plantée ou est inaccessible
    - Ne pourra pas préciser à un GC que l'objet n'est plus référencé
    - Solution : utiliser une durée de bail (*lease*)
    - Si au bout d'un certain temps, un objet distant n'a pas utilisé un objet local, localement, on considère que l'objet distant ne référence plus l'objet local