Middlewares Orientés Messages (MOM)

Basé sur le support de Ada Diaconescu ada.diaconescu@telecom-paristech.fr

Caractéristiques

Un MOM

- Est un système clients/serveurs
- Est un système asynchrone d'échange de messages

  ➔Note: la plupart des implémentations fournissent également une interface synchrone
- Utilise des files de messages
- Possède un « message broker »
- Fournit un API, en général dans plusieurs langages et pour plusieurs systèmes

Utilisations et justifications

Différentes applications ont des besoins et contraintes différents

- **Pas de disponibilité simultané**
  - les différents composants de l'application ne sont pas toujours disponibles
- **Possibilité de communication asynchrone / non-bloquante**
  - La logique métier de l'application permet à un composant d'envoyer des informations à un autre composant et de continuer son exécution
- **Besoins d'un couplage faible**
  - Le développeur veut éviter qu'un composant dépende de l'interface des autres composants ou même de « connaître » les autres composants (Références directes) => remplacement facile

➔ Communication par Message

Principe de fonctionnement !

- **Messagerie inter-application**
  - Asynchrone
  - Non temps réel
  - Offline (Not online)

- **Files de Messages (Message Queueing)**
  - les messages sont mis dans une file d ’attente persistante (i.e. sur disque) avant d ’être relayés vers l’application: guaranteed delivery
  - Partage d ’une file par plusieurs applications Priorité des messages
  - Filtrage des messages à la réception

- **Avantages**
  - Insensible aux partitions de réseaux (sans fil, satellite, WLAN, …)
  - Insensible aux applications non disponibles (temporairement) ou latence

Communication par message !

- **Vue d'ensemble - caractéristiques**
  - Synchrone vs. Asynchrone
  - Persistant vs. Transitoire
  - Unicast vs. Multicast
  - Mode de consommation « push » vs. « pull »
  - Patrons de communication : RPC, Point à Point, Publication / Abonnement, …
  - Systèmes de routage des messages
  - Bibliothèque de traitement de messages
  - …

- **➔ Plusieurs modèles de communication par message**
  - Implantant diverses combinaisons des aspects indiqués auparavant

Communication synchrone vs. asynchrone

- **Synchrone / bloquante**
  - L'émetteur reste bloqué jusqu'à ce que le destinataire acquitte la réception du message (« Acknowledgement »)

- **Asynchrone / non-bloquante**
  - L'émetteur continue de s'exécuter après avoir soumis le message pour la transmission

Données persistantes vs. transitoires

- **Communication persistante (« persistent »)**
  - Le serveur MOM conserve le message jusqu'à ce qu'il soit transmis au récepteur; le message n'est jamais perdu ou effacé ➔ pas de dépendance temporelle : l'émetteur et le récepteur ne sont pas obligés d'être présents en même temps

- **Communication transitoire (« transient »)**
  - Le serveur MOM conserve le message seulement pendant l'exécution simultanée de l'émetteur et du récepteur ➔ dépendance temporelle : l'émetteur et le(s) récepteur(s) doivent être présents en même temps

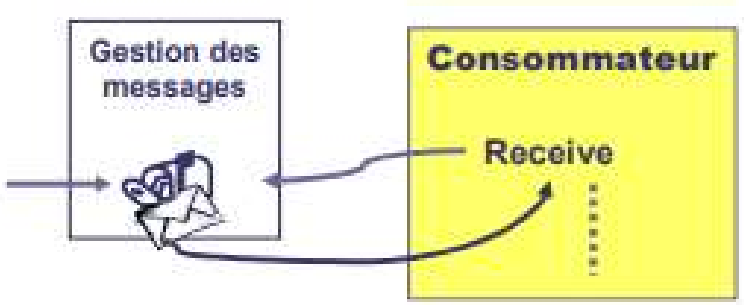
Unicast vs. multicast

- **Unicast**
  - Le message est envoyé à un seul destinataire

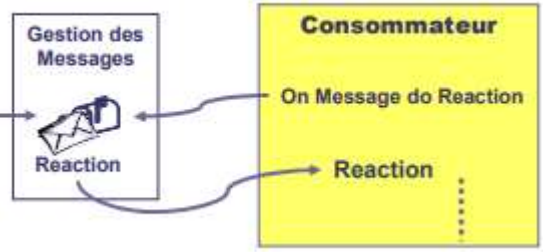
- **Multicast - Diffusion (ou group)**
  - Le message est distribué à plusieurs destinataires

Mode de consommation « push » vs. « pull »

- **Mode « push »**
  - L'émetteur envoie le message au récepte
  - En cas d'absence de message : attente ou exception

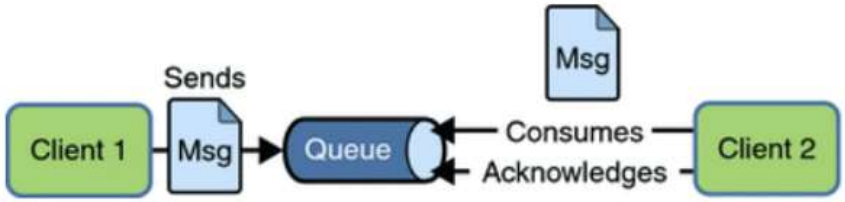


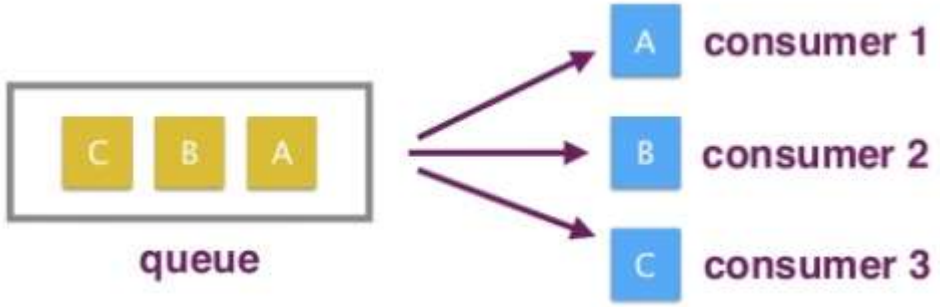
- **Mode « pull »**
  - Le récepteur va chercher le message chez l'émetteur
  - Ex. :
    - périodiquement;
    - en restant bloqué jusqu'à ce qu'un messag devient disponible;
    - sur notification de l'émetteur (« push » et pull »)



Modèle Queue

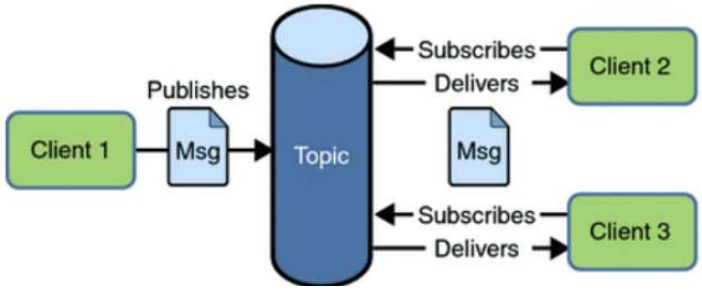
- Chaque message est stocké dans une fille (« Queue ») jusqu'à ce que le destinataire le lise
- Chaque message est consommé une seule fois (par un seul destinataire)
- Pas de dépendance temporelle entre émetteur et destinataire du message
- Le destinataire peut acquitter les messages reçu

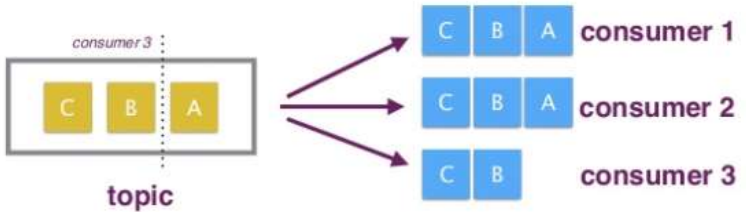




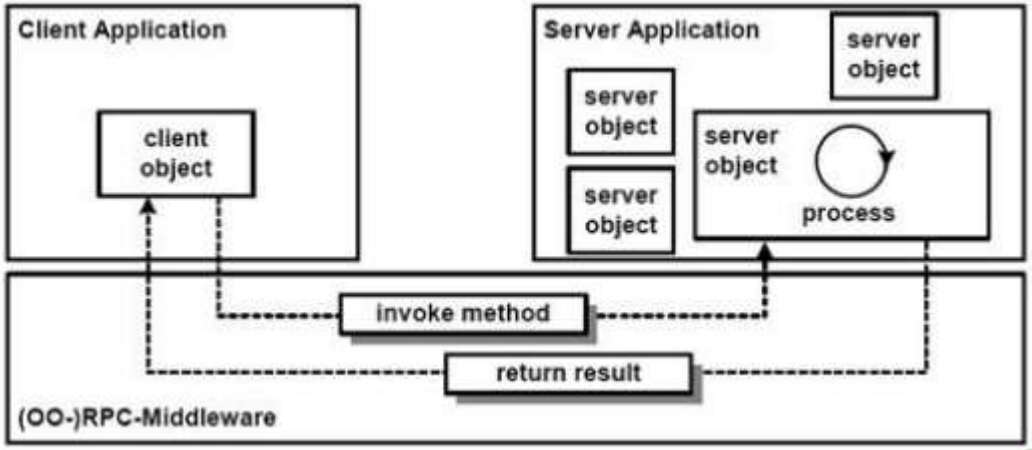
Modèle Topic

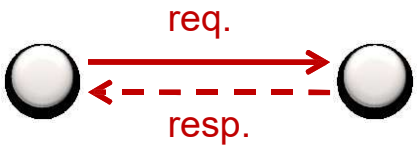
- Le sujet (« Topic ») gère l'envoi de messages pour un ensemble de lecteurs abonnés
- Découplage entre les « publishers » et les « subscribers »
  - Les fournisseurs n'ont pas besoins de connaître les consommateurs
- Dépendance temporelle
  - Un client ne peut lire un message qu'après s''être abonné à un topic,
  - Un client abonné à un topic doit continuer à être actif pour recevoir les message du topic.





Modèle RPC


Client Application
client object
Server Application
server object
server object
server object
server object
process
invoke method
return result
(OO-)RPC-Middleware


req.
resp.

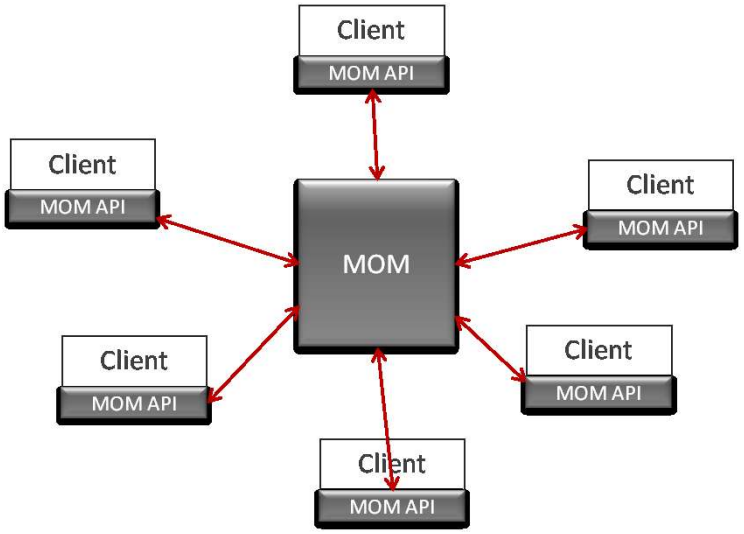
Topologies MOM

- Différentes topologies possibles pour le Message Broker
  - Centralisée
  - Décentralisée
  - Hybride

- Chaque produit MOM peut utiliser une ou plusieurs topologies

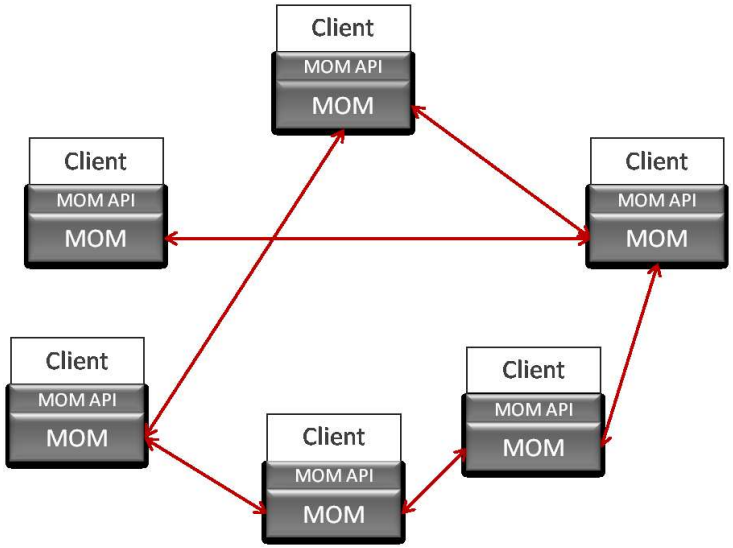
MOM – topologie centralisée (« hub & spoke »)

- **Serveur central de gestion de messages**
  - Routage, stockage, transactions, ..
- **Avantages**
  - Découple les clients, qui ne voient que le serveur - l'ajout ou l'enlèvement d'un client n'a pas d'impact sur les autres clients
- **Inconvénients**
  - Engendre un trafic réseau important
  - Crée un goulot d'étranglement:
    - Point unique d'échec(« unique point of failure »)
    - Problèmes de passage à l'échelle
    - Possibilité de performance réduite (latence)



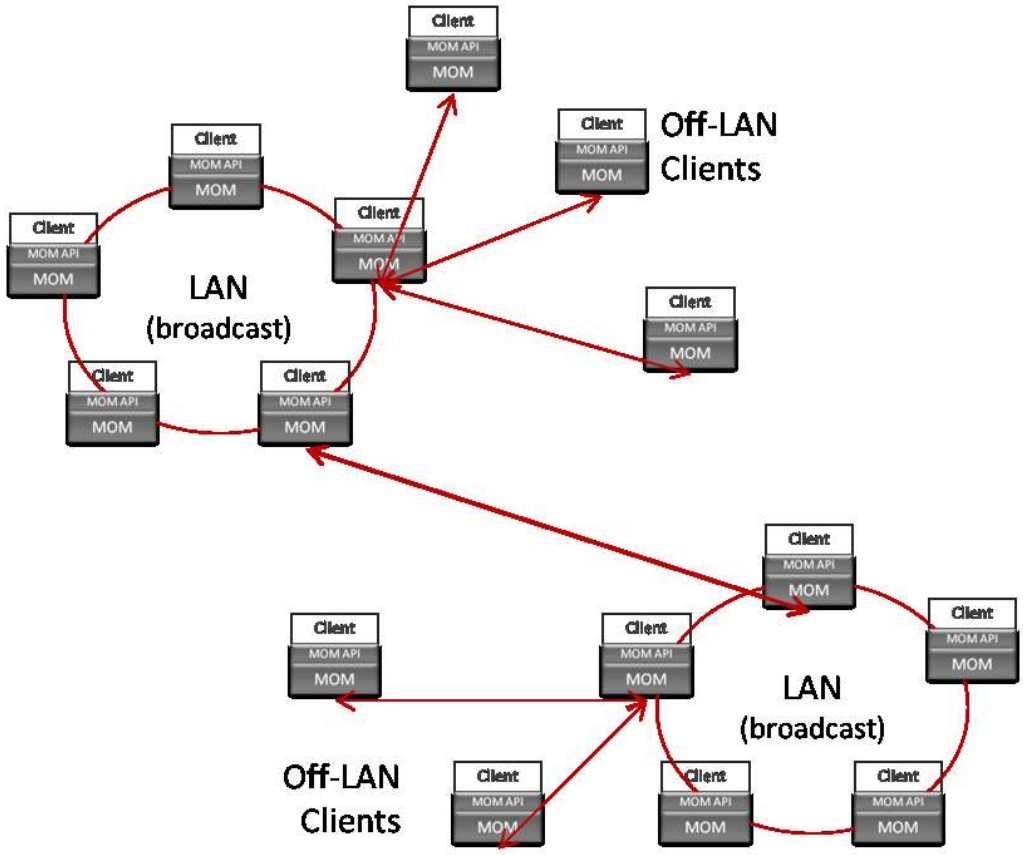
MOM – topologie décentralisée

- **Une instance de MOM est installé chez chaque Client - pas de serveur central**
  - Stockage, transactions, sécurité, …
  - Routage de messages
    - Ad-hoc – entre les MOMs
    - Basé sur le protocole réseaux existant - ex. IP-Multicast
- **Avantages**
  - Distribution des fonctions du MOM entre les serveurs – ex. persistance, sécurité, …
- **Inconvénients**
  - Problèmes potentiels d'interopérabilité – ex. différents vendeurs ou versions de MOM
  - Clients plus lourds, duplication des fonctions



MOM – topologie hybride

- Combine les deux architectures précédentes
- Combine les avantages et les inconvénients



Sun JMS: Java Message Service

- **Spécification d‟un API pour les MOM**
- **Intégré à J2EE 1.3 ++, couplage avec les EJB (Message-driven bean)**
- **Première spécification d‟un MOM publiquement accessible**
  - Implémentée par les principaux MOM
  - Adaptable à d'autres langages (C++, Ada)
  - Peu restrictive: synthèse des MOM existants => autorise plutôt qu'interdit

Réception de messages JMS

- **Attention à l'utilisation des termes « synchrone » & « asynchrone »!**

- Réception **Synchrone -** mode « pull », bloquant
  - Le consommateur récupère explicitement le message depuis la destination en appelant la méthode **receive**.
  - Cette méthode bloque jusqu'à ce qu'un message soit disponible, ou qu'un délai expire.

- Réception **Asynchrone – mode « push »,** avec dépendance temporelle
  - Le consommateur enregistre un « message listener » auprès de la Destination ciblée
  - Lorsqu'un message arrive, le fournisseur JMS délivre le message au « message listener » en appelant la méthode **onMessage**

Types de Destinations JMS

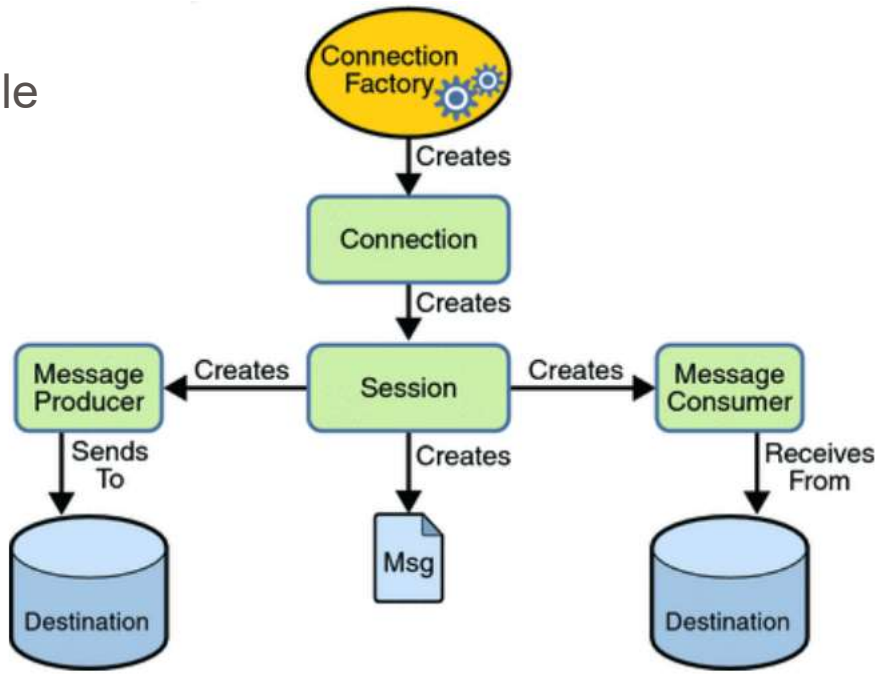
- **File - « Queue »**
  - Persistance des messages
  - Découplage temporel entre le producteur et le consommateur des messages
  - Habituellement utilisé pour la communication Point-à-Point

- **Sujet - « Topic »**
  - Non-persistance des messages
  - Couplage temporel entre le producteur et le consommateur des messages (sauf utilisation de l'option « durable subscription »)
  - Habituellement utilisé pour la communication Publication/Abonnement

- *Note: JMS permet l'utilisation des deux types de destinations - queues et topics, avec les deux modes de réception - synchrone et asynchrone.*

JMS Client –
Comment initialiser la communication ?

- **Une Usine de Connexions –**
  « Connection Factory »
  - Prend en charge la connexion avec le fournisseur JMS (MOM).
  - Encapsule les paramètres de connexion mis en place par l'"administrateur.



- **Une Session**
  - Est un contexte mono-tache
  - Utilisé pour l'émission et la réception de messages
  - Gère plusieurs consommateurs et producteurs de message

JMS - - Lien avec d'autres API Java

- JNDI: annuaire
- JTA: transaction