Moyens de communication Interprocessus

- ***Introduction***
- Représentation de données dans plusieurs plateformes
- Plusieurs modes de communication:
  - Communication par messages : Socket
  - Communication par procédure à distance: RPC
  - Communication par invocation d'objet à distance
    - CORBA
    - RMI-Java
  - Communication par événements et par flot
- Lecture et Références

Introduction

- Les programmeurs du système doivent connaître
  - Le nom des opérations que le serveur comprenne
  - Le nombre et le format des paramètres
    - Ils doivent coder la conversion des paramètres eux-mêmes
    - Ces informations définissent un **protocole** entre client et serveur
      - Sans connaissance du protocole, les processus ne peuvent pas se communiquer !
- Moyens de communication
  - La procédure est un moyen simple et efficace pour la programmation séquentielle. Peut-on l'utiliser dans le contexte distribué ?
  - La procédure, la méthode (objet), le message, l'évènement et le flot (stream) sont des moyens pour la programmation distribuée!

Techniques de communication de Middleware

- Remote Procedure Call
- Message-Oriented Communication
- Stream-Oriented Communication
- Multicast Communication

Types de Communication

- Persistant versus transitoire (transient)
- Synchrone versus asynchrone
- Discrète versus streaming

Persistent versus Transient Communication

- **Persistent**: messages are held by the middleware comm. service until they can be delivered (e.g., email)
  - Sender can terminate after executing send
  - Receiver will get message next time it runs
- **Transient**: messages exist only while the sender and receiver are running
  - Communication errors or inactive receiver cause the message to be discarded
  - Transport-level communication is transient
- **RPC?**

Asynchronous v Synchronous Communication [Tanenbaum 2007]

- **Asynchronous**: (non-blocking) sender resumes execution as soon as the message is passed to the communication/middleware software

- **Synchronous**: sender is blocked until
  - The OS or middleware notifies acceptance of the message, *or*
  - The message has been delivered to the receiver, *or*
  - The receiver processes it & returns a response

Communication synchrone et asynchrone

- **Caractéristiques de la communication inter-processus :**
  - **Synchrone** : les processus se synchronisent à chaque envoi de messages

    Send et Receive : opérations bloquantes
  - **Asynchrone** : pas de synchronisation entre émetteur et récepteur

    Send : opération non-bloquante
    Receive : opération bloquante ou non

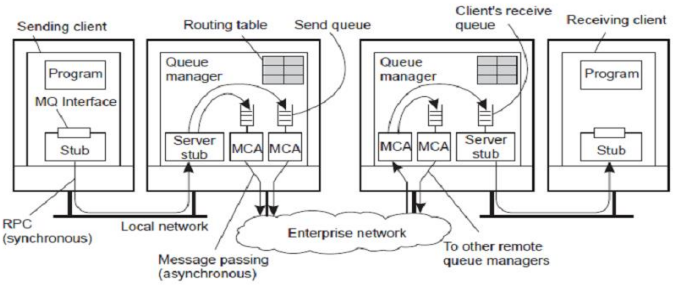
Discrete versus Streaming Communication

- **Discrete**: communicating parties exchange discrete messages
- **Streaming**: one-way communication; a "session" consists of multiple messages from the sender that are related either by send order (TCP streams), temporal proximity (multimedia streams), etc.

Message-Oriented Middleware (MOM) - Persistent

- Processes communicate through message queues
  - *Queues are maintained by the message-queuing system*
  - Sender appends to queue, receiver removes from queue
  - Neither the sender nor receiver needs to be on-line when the message is transmitted

IBM's WebSphere MQ



Stream-Oriented Communication

- RPC and message-oriented communication are based on the exchange of ***discrete*** messages
  - Timing might affect performance, but not correctness

- In stream-oriented communication the message content (multimedia streams) **must be delivered at a certain rate**, as well as correctly
  - e.g., music or video

Data Streams

- **Asynchronous transmission mode:** the order is important, and data is transmitted one after the other, no restriction to when data is to be delivered
- **Synchronous transmission mode** defines a maximum end-to-end delay for individual data packets
- **Isochronous transmission mode** has a maximum and minimum end-to-end delay requirement (jitter is bounded)
  - Not too slow, but not too fast either

Stream

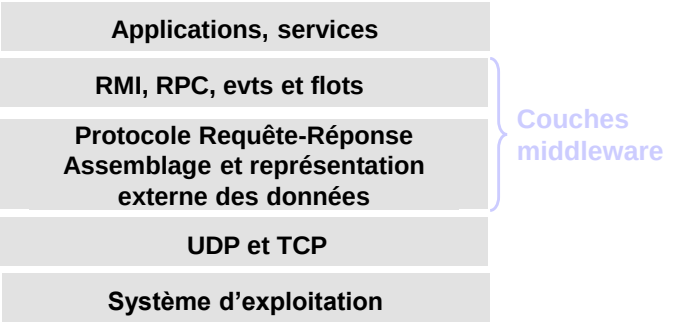
Definition

A (continuous) data stream is a connection-oriented communication facility that supports isochronous data transmission.

Some common stream characteristics

- Streams are unidirectional
- There is generally a single source, and one or more sinks
- Often, either the sink and/or source is a wrapper around hardware (e.g., camera, CD device, TV monitor)
- Simple stream: a single flow of data, e.g., audio or video
- Complex stream: multiple data flows, e.g., stereo audio or combination audio/video

Couches d'un middleware



Caractéristiques d'un middleware

- **Caractéristiques des couches middleware :**
  - **Transparence** (définition)
  - **Indépendantes** : les protocoles de communication supportant les abstractions middleware doivent être indépendants des protocoles de transport (exemple)
  - **Matériels hétérogènes** : masquer les différences entre les architectures matérielles (big-endian, litte-endian, …) (moyen)
  - **Plusieurs langages de programmation** : permettre à une application répartie d'utiliser plusieurs langages de programmation (exemple)

Exemples des middlewares

RPC/IDL :

```
program RAND_PROG
{
  version RAND_VERS
  {
     void INITIALIZE_RANDOM(long) = 1;
     double GET_NEXT_RANDOM(void) = 2;
   } = 1;
} = 0x31111111;
```

CORBA/IDL :

```
module EXAMPLES
{
  interface RAND_PROG
  {
    void INITIALIZE_RANDOM(in long arg);
    double GET_NEXT_RANDOM(void);
  };
};
```

Définition d'une procédure distante avec différents IDL

RMI/INTERFACE :

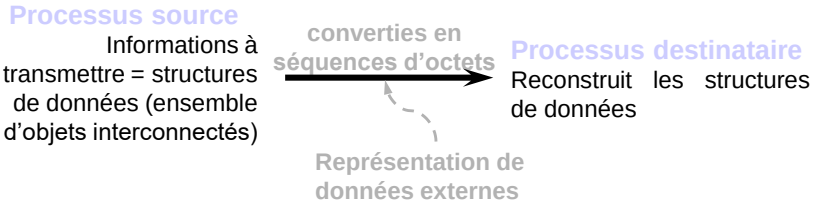
```
Import java.rmi.*;
package EXAMPLE;
public interface RAND_VERS extends Remote
{
    public void INITIALIZE_RANDOM(long arg) throws RemoteExceptions;
    public double GET_NEXT_RANDOM(void) throws RemoteExceptions;
};
```


Moyens de communication Interprocessus

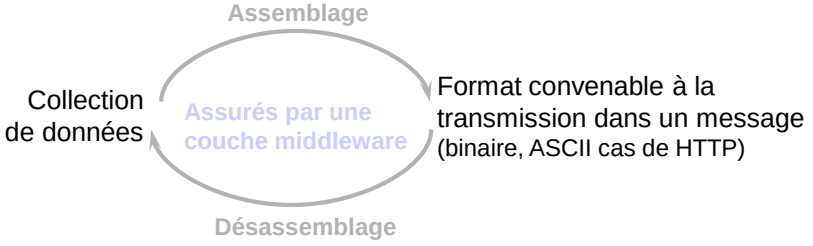
- Introduction
- ***Représentation de données dans plusieurs plateformes***
- Plusieurs modes de communication:
  - Communication par messages : Socket
  - Communication par procédure à distance: RPC
  - Communication par invocation d'objet à distance
    - CORBA
    - RMI-Java
  - Communication par événements et par flot
- Lecture et Références

Représentation des données et Codage

- **Différentes représentations :**
  - Virgule flottante représentée de différentes façons
  - Différents codages pour représenter les caractères
  - Différents formats : Little-endian, big-endian



Représentation des données et Codage



- **Différentes représentations :**
  - **CDR de CORBA (Common Data Representation)**
  - **Sérialisation d'objets JAVA**
  - **Standard XDR de SUN :** échange de messages entre clients et serveurs SUN NFS (Network File System)

CDR de CORBA

- Définit avec CORBA 2.0 [Object Management Group, 1998]
- Format binaire : arguments et résultats des invocations
- Peut représenter tous les types de données : short, long, unsigned short, unsigned long, float, double, char, boolean, octet, any (tout type simple ou composé : string, array, struct, enumerated, union, …)
- CDR (CORBA) & XDR (SUN) : types non inclus, supposent que l'émetteur et le récepteur connaissent l'ordre et le types des champs

CDR de CORBA

- **Exemple :**

{'Smith', 'London', 1934} → struct Person { string name;
string place;
long year;};

index in sequence of bytes	←4 bytes→	*notes on representation*
0–3	5	*length of string*
4–7	"Smit"	*'Smith'*
8–11	"h___"	
12–15	6	*length of string*
16–19	"Lond"	*'London'*
20–23	"on__"	
24–27	1934	*unsigned long*

unsigned long (32 bits)

Sérialisation d'objets Java

- Java RMI : objets passés comme paramètres ou résultats à des invocations de méthodes
- Format binaire : message construit à partir d'un objet ou une hiérarchie d'objets
- Classe **ObjectOutputStream** : écriture du contenu des instances des variables

  Classe **ObjectInputStream** : Lecture

Sérialisation d'objets Java

- **Exemple :** Classe Person

```
Public class Person implements Serializable {
   private String name;
   private String place;
   private int year;
   public Person (String aName,
                  String a Place, int aYear) {
         name = aName;
         place = aPlace;
         year = aYear;
   } … }
```

```
Person p= new Person("Smith",
                     "London",
                     1934);
```

Serialized values				*Explanation*
Person	8-byte version number		h0	*class name, version number*
3	int year	java.lang.String name:	java.lang.String place:	*number, type and name of instance variables*
1934	5 Smith	6 London	h1	*values of instance variables*

Sérialisation d'objets Java

- Classe **ObjectOutputStream**

- **ObjectOutputStream**(OutputStream)
- **ObjectOutputStream**(void)
- **write**(byte[])
- **write**(int)
- **writeByte**(int)
- **writeChar**(int)
- **writeFloat**(float)
- **writeLong**(long)
- **writeShort**(int)
- **writeUTF**(String)
- **writeBytes**(String)
- **writeObject**(Object)
- **writeDouble**(double)
- **writeBoolean**(boolean)
- **…**

```
Person p= new Person("Smith", "London", 1934);

// Sérialisation
ObjectOutputStream SP = new ObjectOutputStream();
SP.writeObject(p);

// Désérialisation
ObjectInputStream DSP = new ObjectInputStream();
Person = DSP.readObject();
```


Référence à un objet distant

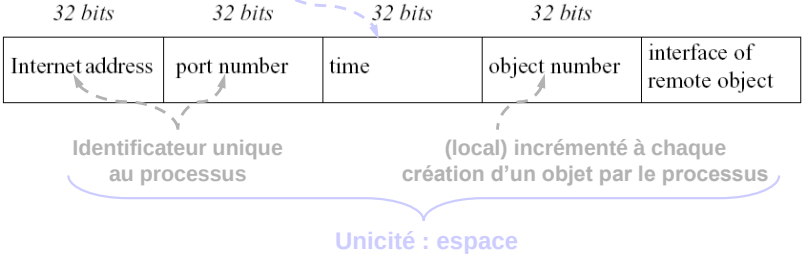
Processus Client
Invoque la méthode d'un objet distant

Message d'invocation →

Processus Serveur

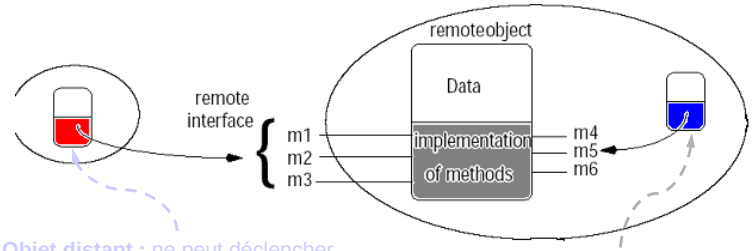
Référence à un Object Distant :
identificateur d'un objet distant valide dans un système distribué

Unicité : temps



Interface distante

- Spécifie quelle méthode peut être invoquée à distance



Objet distant : ne peut déclencher que les méthodes m1-m3 publiées dans l'interface distante

Objet local : peut déclencher toutes les méthodes m1-m6

Interface distante

- **Exemple :** CORBA IDL (Interface Definition Langage)

```
// Dans fichier Person.idl
struct Person {
        string name;
        string place;
        long year;
} ;
interface PersonList {
        readonly attribute string listname;
        void addPerson(in Person p) ;
        void getPerson(in string name, out Person p);
        long number();
};
```

RMI (Remote Method Invocation)

- **Sémantiques de RMI :**

Mesures de tolérance aux fautes

Retransmettre requête	Filtrage des doubles	Réexécuter procédure ou retransmettre réponse	Sémantiques des invocations
Non	Non applicable	Non applicable	***Peut-être*** **CORBA**
Oui	Non	Réexécuter procédure	***Au-Moins-Une*** **SUN/RPC**
Oui	Oui	retransmettre réponse	***Au-Plus-Une*** **Java RMI, CORBA**

Extensible Markup Language XML

- XML permet aux clients de communiquer avec les services web: SOAP
- XML : définir les interfaces et autres propriétés des services web
- Format textuel: message construit en utilisant de balises (tags) à partir d'un objet (élément) ou une hiérarchie d'objets
- XML: transférer l'information et son type : contrairement à CORBA-CDR: car XML a multiple utilisations
- Exemple: définition XML de la structure Person:

```
<person id="123456789">
  <name>Smith</name>
  <place>London</place>
  <year>1934</year>
  <!- a comment -->
</person>
```

Moyens de communication Interprocessus

- Introduction
- Protocoles en couche
- Représentation de données dans plusieurs plateformes
- ***Plusieurs modes de communication:***
  - ***Communication par messages : Socket***
  - Communication par procédure à distance: RPC
  - Communication par invocation d'objet à distance
    - CORBA
    - RMI-Java
  - Communication par événements et par flot
- Références

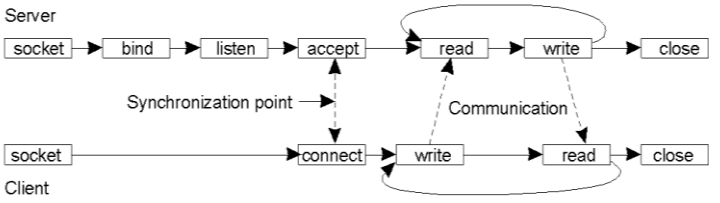
Communication par message

- La façon naturelle de réaliser la communication en RPC et ROI (Remote Object Invocation) est la synchronisation!
  - RPC et appel d'objet distant :
    - Grande transparence
    - Communication synchrone
    - Exige une réponse immédiate du serveur.
- Communication par messages :
  - 1er cas : toutes les parties sont en opération
  - 2ième cas : passage de message avec file d'attente, les parties n'ont pas à être en opération simultanément.

Socket Berkeley

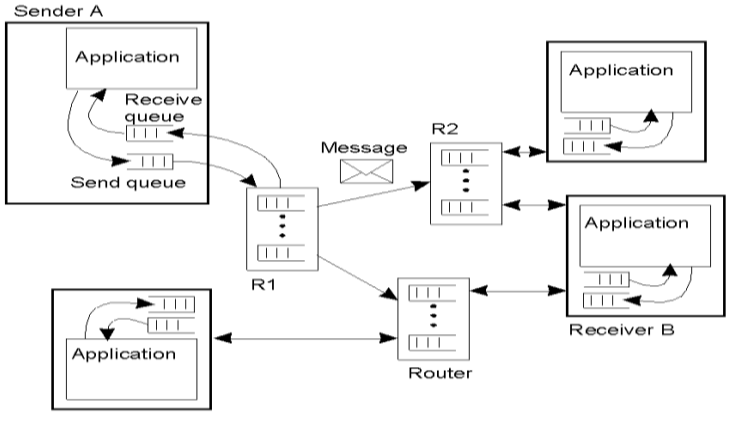
primitives de socket pour TCP/IP

Primitive	Meaning
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection



- Mode orienté connexion utilisé par les sockets (TCP).

Systèmes avec mise en file d'attente et routeurs



- Avantage : les applications n'ont pas à connaître la topologie du réseau, seul les routeurs doivent la connaître, partiellement ou en entier.

Communication par message

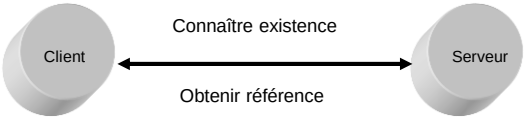
- Avantages et inconvénients des styles synchrones et asynchrones
  - Mode synchrone (bloquant)
    - Terminant – la fin est certaine
    - Avec acquittement – réponse
    - Pb: arrêt complet du fonctionnement du client quand le serveur ne répond pas
  - Mode asynchrone (non bloquant)
    - Pour acquittement et terminaison, il faut un protocole
    - Pb de gestion des rythmes entre serveurs et clients

Moyens de communication Interprocessus

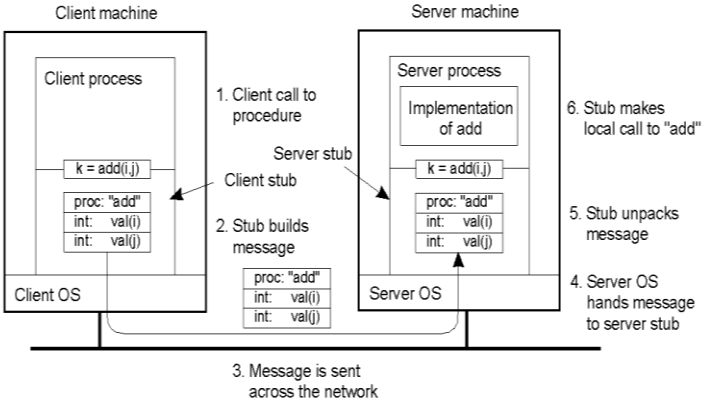
- Introduction
- Protocoles en couche
- Représentation de données dans plusieurs plateformes
- ***Plusieurs modes de communication:***
  - Communication par messages : Socket
  - ***Communication par procédure à distance: RPC***
  - Communication par invocation d'objet à distance
    - CORBA
    - RMI-Java
  - Communication par événements et par flot
- Références

Communication par procédure à distance

- Plusieurs systèmes distribués sont basés sur l'échange de message explicitement. Ceci ne supporte pas l'accès transparent.
- Birell et Nelson 1984, ont introduit une façon complètement différente pour traiter la communication:
- Appel de procédure à distance :
  - *A* appelle une procédure sur *B*;
  - *A* est suspendu et la procédure est exécutée sur *B*;
  - Lorsque la procédure est terminée sur *B*, le résultat est retourné à *A*, qui continue son exécution.
- Complication : paramètres et résultats doivent être passés entre différentes machines hétérogènes.



RPC et passages de paramètres



- Étapes effectuées en faisant appel à une procédure à distance avec RPC

Considérations avec le passage par référence

- Pointeur comme paramètre : contenu de la mémoire d'une station *A* *ne* pas le même sur la station *B*
- Première solution : interdire la passage par référence (très limitatif!)
- Lorsque le pointeur pointe à un tableau de dimension connue, transmettre le tableau entre client au serveur (passage par **copie/restauration**)
- Éviter les copies inutiles : spécifier les directions (**in/out**) de la transmission du tableau
- Passage de structures plus complexes (graphes, etc.) : impossible dans le cas général.

Spécification des paramètres pour la génération des souches

a) une procedure
b) Le message correspondant

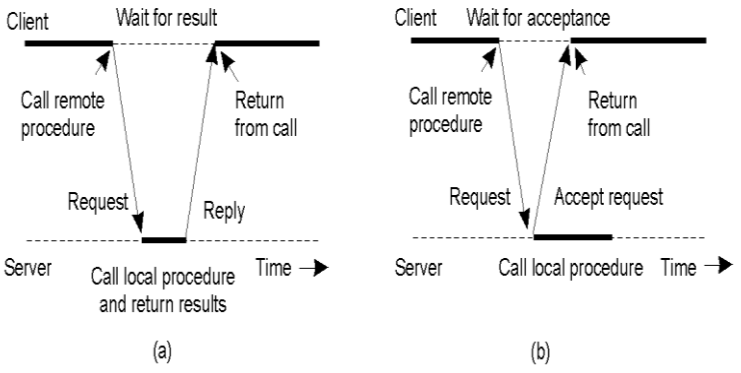
```
foobar( char x; float y; int z[5] )
{
  ....
}
```

(a)

foobar's local variables	
	x
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

(b)

RPC asynchrone



(a) RPC synchrone (b) RPC asynchrone

Communication par procédure à distance

- Amélioration et simplification de la structuration des applications reparties par rapport au mode message
- Mais beaucoup de difficultés :
  - Syntaxe plus lourde qu'en local
  - Sémantique différente
    - Performances
    - Transmission des arguments
    - Mode de pannes
    - Sécurité
- Outils de développement limités à la génération automatique des souches – pas de déploiement -
- Exemple : RPC sun, RPC DCE, etc.

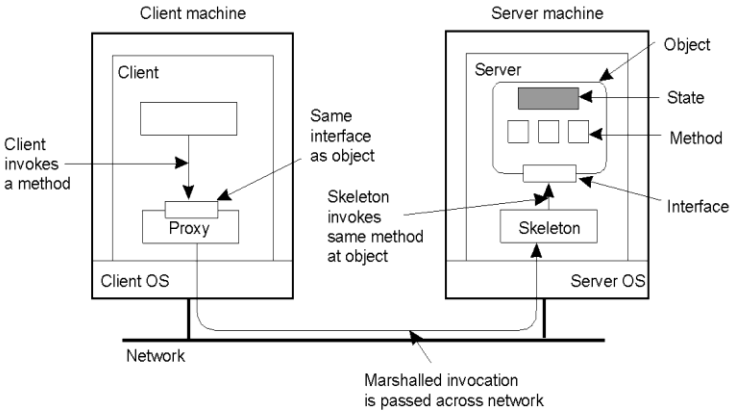
Moyens de communication Interprocessus

- Introduction
- Protocoles en couche
- Représentation de données dans plusieurs plateformes
- ***Plusieurs modes de communication:***
  - Communication par messages : Socket
  - Communication par procédure à distance: RPC
  - ***Communication par invocation d'objet à distance***
    - CORBA
    - RMI-Java
  - Communication par événements et par flot
- Références

Communication par objet

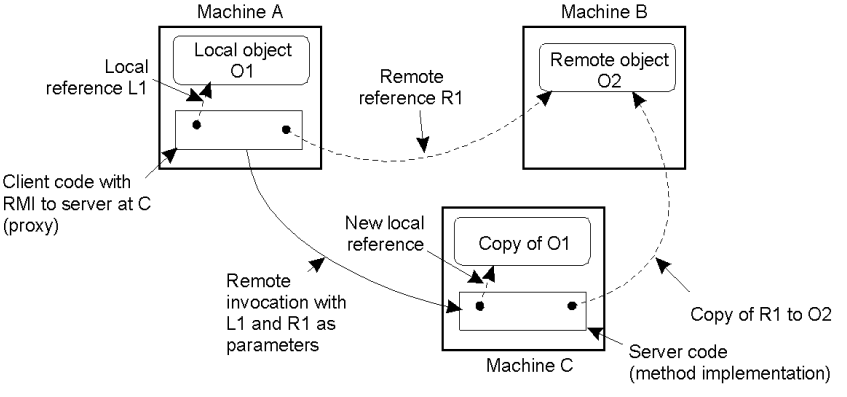
- Objets distribués :
  - Objet = état interne + méthodes;
  - Méthodes disponibles via des interfaces;
  - Séparation entre les interfaces et l'implémentation: améliore la transparence
  - Objet distribué : Une séparation stricte nous permet de placer une interface sur une machine tandis que l'objet lui-même réside sur une autre machine.
  - Les paramètres sont passés d'une façon transparente aux objets distants.
  - Exemple : RMI-Java, Corba, DCOM, EJB, etc.
  - RMI-Java garantit :
    - des mécanismes dynamiques de parallélisme (par le biais des processus légers) et
    - une vraie portabilité (par le biais de la compilation pour une machine virtuelle Java)

Objets appelés à distance



- Organisation commune d'un objet distant avec proxy

Passage de paramètres



La situation quand un objet passé par reférence ou par valeur.

D'autres modèles de programmation

- La migration favorise les applications qui consomment beaucoup des données
  - P.ex: les calculs scientifiques
    - Puisqu'on minimalise l'utilisation du réseau
  - Mode d'opération déconnecté
    - Un utilisateur peut charger son travail sur le réseau, se déconnecter et puis se connecter plus tard pour avoir ses résultats
      - P.ex: la recherche des documents en-ligne
      - Idéal pour l'Internet puisque la connexion est coûteuse
  - Exemples :
    - Les servlets, les agents mobiles

Moyens de communication Interprocessus

- Introduction
- Protocoles en couche
- Représentation de données dans plusieurs plateformes
- ***Plusieurs modes de communication:***
  - Communication par messages : Socket
  - Communication par procédure à distance: RPC
  - Communication par invocation d'objet à distance
    - CORBA
    - RMI-Java
  - ***Communication par événements et par flot***
- Références

Programmation par Évènements (1)

- Permettre à des objets de recevoir des notifications d'événements survenus dans d'autres objets
- **Notification d'événements :** asynchrone, récepteur détermine le traitement approprié (exemple)
- Utilisation du paradigme : **publish-subscribe** (explication)
- **Types d'événements :**

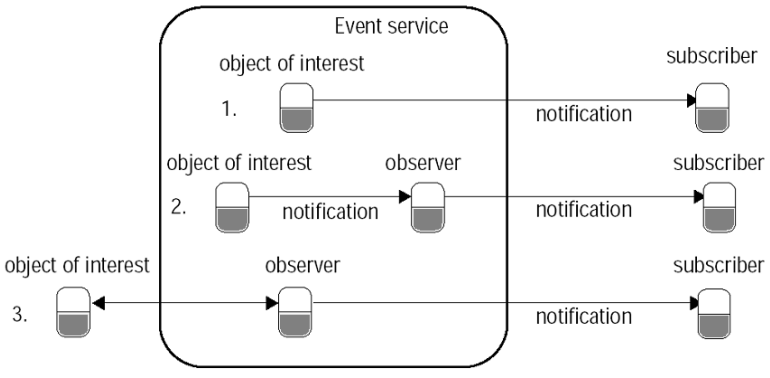
Source d'événements —génère→ Événements de différents types

Chaque événement a des attributs :
- Nom ou identificateur de l'objet
- Opération
- Ses paramètres
- Temps, …

- Types et attributs : utilisés pour s'inscrire (s'abonner) à des notifications d'événements

Programmation par Évènements (2)

- **Participants de la notification d'événements distribués** (détails) **:**



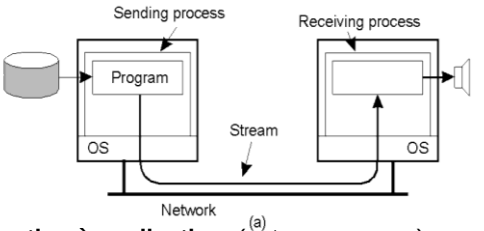
Communication par flot

- Information avec relation temporelle : audio, vidéo, etc.
- Flots de données : séquence d'unités de données
- Trois modes de transmission :
  - Mode asynchrone : unités de données transmises une après l'autre, sans autre contrainte temporelle;
  - Mode synchrone : délais d'expiration de chaque unité de données;
  - Mode isochrone : délais minimum et maximum de transmission de chaque unité de données => ce qui est généralement utilisé pour transmettre de l'audio et du vidéo

Flot simple et flot complexe

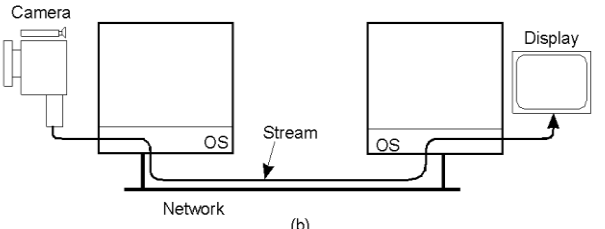
- Flot simple : une seule séquence de données
- Flot complexe : regroupement de plusieurs flots simples relatés, des sous-flots :
  - Exemple : trame audio stéréo, films, sous-titrage
  - Dépendance temporelle entre les sous-flots d'un même flot complexe
  - Flot application à application ou flot ressource à ressource
  - Communication multi-tiers (multidiffusion).

Flot application à application et flot ressource à ressource



(a)

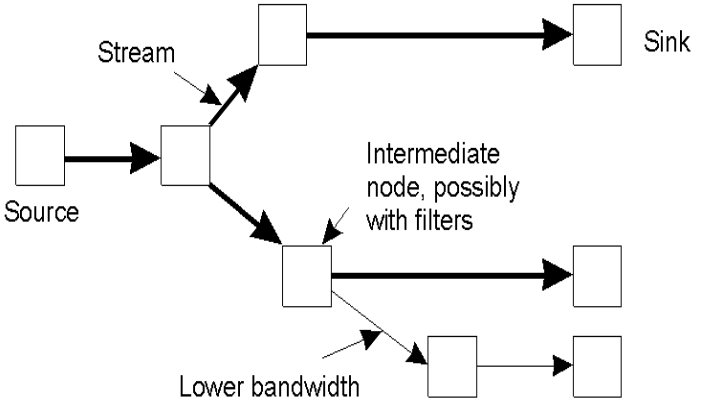
- (a) **Flot application à application** (entre processus)



(b)

- **(b) Flot ressource à ressource**

Flot avec multidiffusion

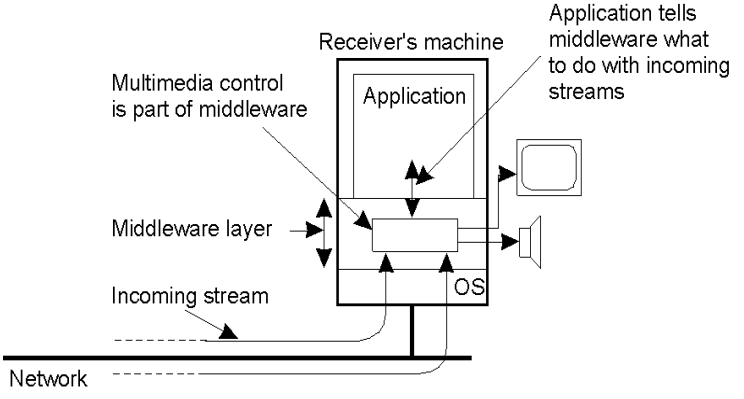


- An example of multicasting a stream to several receivers.

Synchronisation des flots

- Synchronisation implicite d'information continue et discrète : exemple de présentation avec acétates augmentées de propos audio
- Trame audio stéréo : synchroniser les canaux gauche et droite
- Films : synchronisation de l'audio avec le vidéo
- MPEG-2 : flot découpé en paquets avec identifiant temporel (donné à 90 kHz) et multiplexé dans un flot spécifique au programme (synchronisation du côté expéditeur).

Synchronisation du côté destinataire



Références

- Lectures :
  - **Chapitre 4 et 5** du livre Coulouris, 2011.
  - **Chapitre 2** du livre Steen et Tanenbaum, 2017
- Coulouris, G. et al. 2011. Distributed systems, concepts and design.
- Tanenbaum, 2017. Distributed System: principles and paradgms, third edition.
- Les cours de Laurence Duchien
- arcad.essi.fr/2002-10-composant/slides/12-budau.ppt
- acétates de M. Abdul Obaid.
- acétates de M. Christian Gagné.