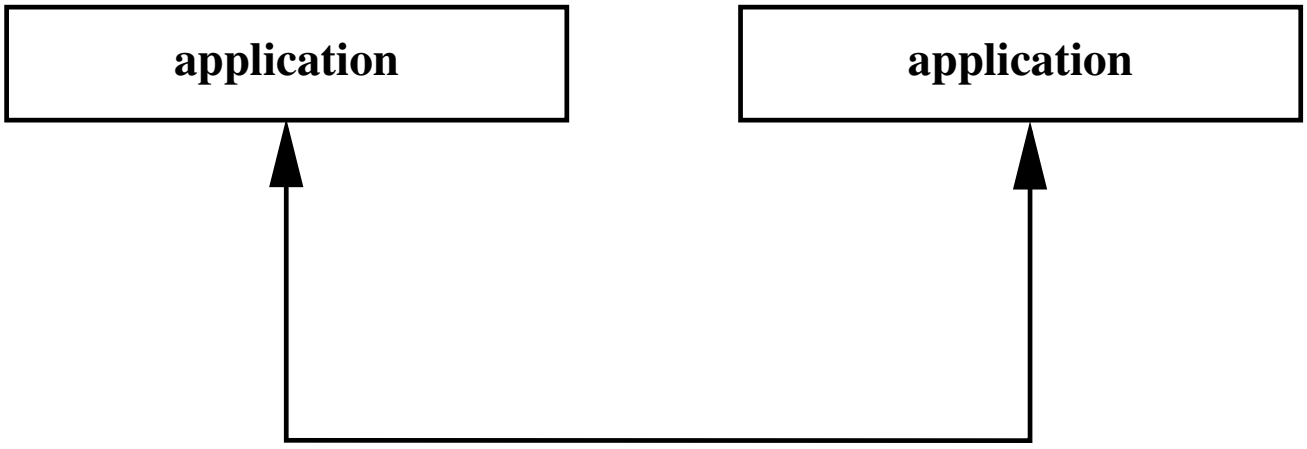
Middleware

Le 12 septembre 2012 , SVN-ID 231

Table des matières

Problème

2 entités s'échangent des données:



Codeur/décodeur petit/grand indien, ASCII/EBCDIC, flottant, alignements, langages $\longrightarrow$ PDU obligatoires

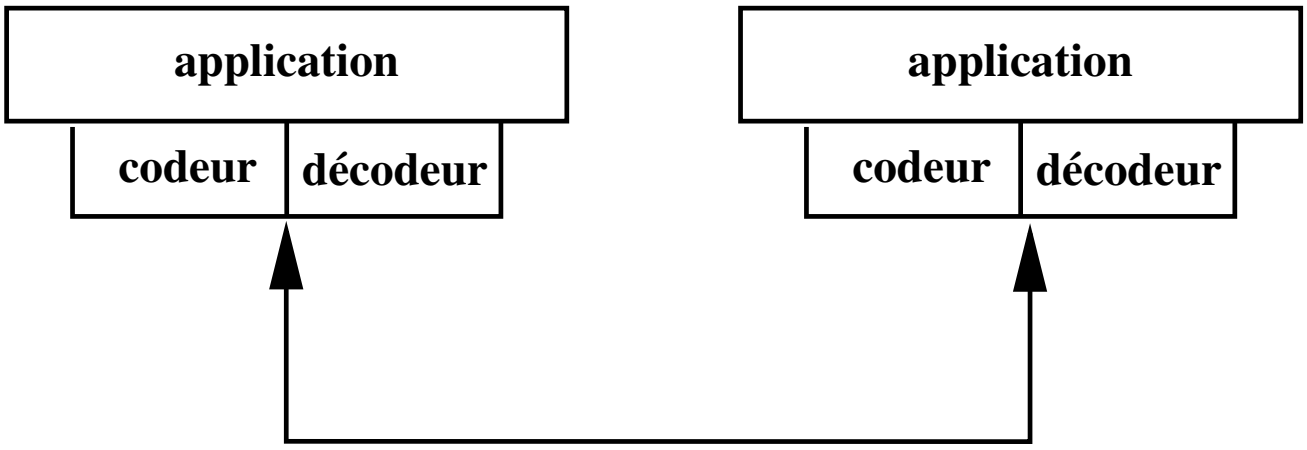
Transmission supports physiques de types et de qualités variés (timeout, retransmission, code détecteur d'erreur)

$\Longrightarrow$ nombre d'occurrences élevé: M x L [x S] avec M:type de machine ; L:langage ; S:support

exemple: 24 avec M=4 (386 32/64bit Unix/Windows), L=6 (C, C++, java, python, php, basic)

Problème

2 entités s'échangent des données:



Codeur/décodeur petit/grand indien, ASCII/EBCDIC, flottant, alignements, langages $\longrightarrow$ PDU obligatoires

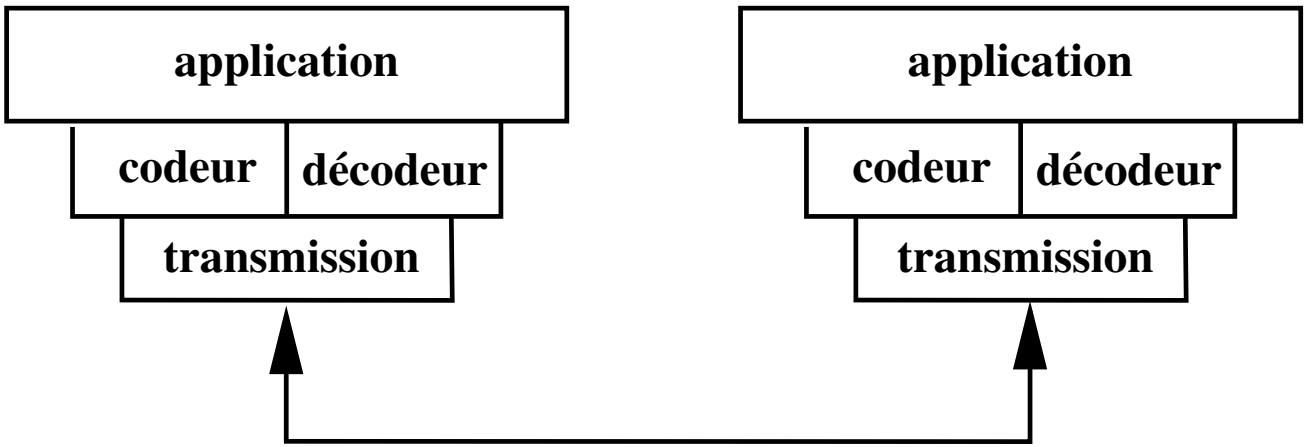
Transmission supports physiques de types et de qualités variés (timeout, retransmission, code détecteur d'erreur)

$\Longrightarrow$ nombre d'occurrences élevé: M x L [x S] avec M:type de machine ; L:langage ; S:support

exemple: 24 avec M=4 (386 32/64bit Unix/Windows), L=6 (C, C++, java, python, php, basic)

Problème

2 entités s'échangent des données:



Codeur/décodeur petit/grand indien, ASCII/EBCDIC, flottant, alignements, langages $\longrightarrow$ PDU obligatoires

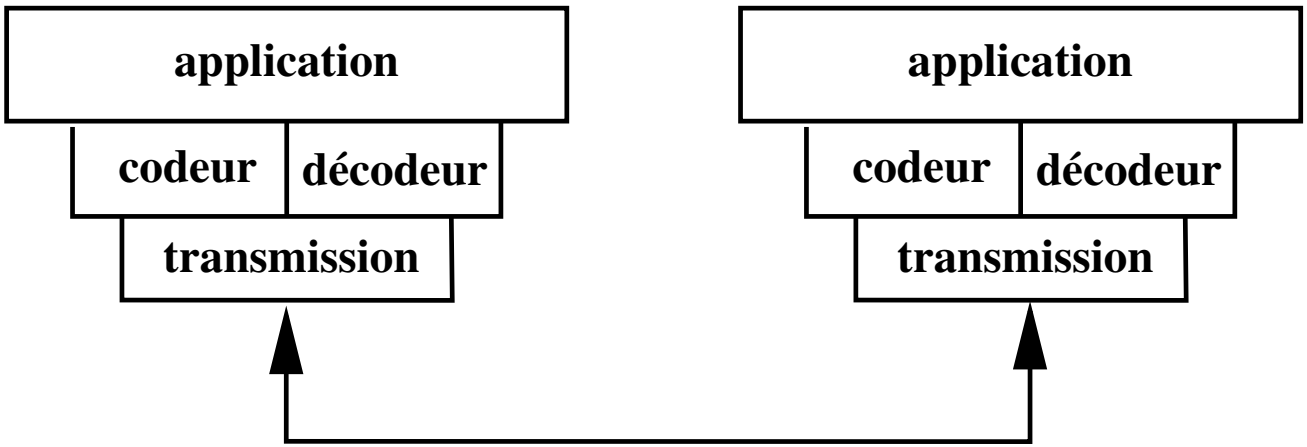
Transmission supports physiques de types et de qualités variés (timeout, retransmission, code détecteur d'erreur)

$\Longrightarrow$ nombre d'occurrences élevé: M x L [x S] avec M:type de machine ; L:langage ; S:support

exemple: 24 avec M=4 (386 32/64bit Unix/Windows), L=6 (C, C++, java, python, php, basic)

Problème

2 entités s'échangent des données:



Codeur/décodeur petit/grand indien, ASCII/EBCDIC, flottant, alignements, langages $\longrightarrow$ PDU obligatoires

Transmission supports physiques de types et de qualités variés (timeout, retransmission, code détecteur d'erreur)

$\Longrightarrow$ nombre d'occurrences élevé: M x L [x S] avec M:type de machine ; L:langage ; S:support

exemple: 24 avec M=4 (386 32/64bit Unix/Windows), L=6 (C, C++, java, python, php, basic)

Exemple: Mini-serveur financier

les indices: CAC40, DOW JONES, NIKKEI, ...

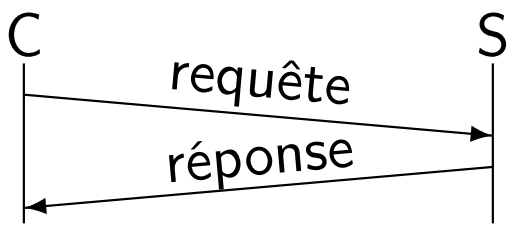
les services:

1) obtention de la valeur du jour d'un indice.
2) obtention des valeurs des 10 derniers jours d'un indice.

modèle réseau:

protocole:

question/réponse



services 1

- PDU_0 requête: getcur, indice
- PDU_1 reponse: status, réel

services 2

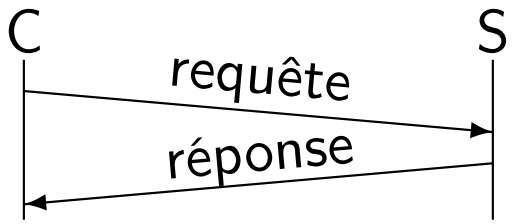
- DU_2 requête: getlast, indice
- DU_3 reponse: status, nb, $\{réel_0, réel_1, ...\}$

Exemple: Mini-serveur financier

modèle réseau:

protocole:

question/réponse



services 1

- PDU_0 requête: getcur, indice
- PDU_1 reponse: status, réel

services 2

- DU_2 requête: getlast, indice
- DU_3 reponse: status, nb, {$réel_0$, $réel_1$, ...}

définition des 4 PDUs

octets	0	1 2	3		....
	C	ID	indice (10)		
	L	ID	indice (10)		
	R	ID	statux (1)	valeur (4)	
	M	ID	statux (1)	nb (1)	valeur (4*nb)

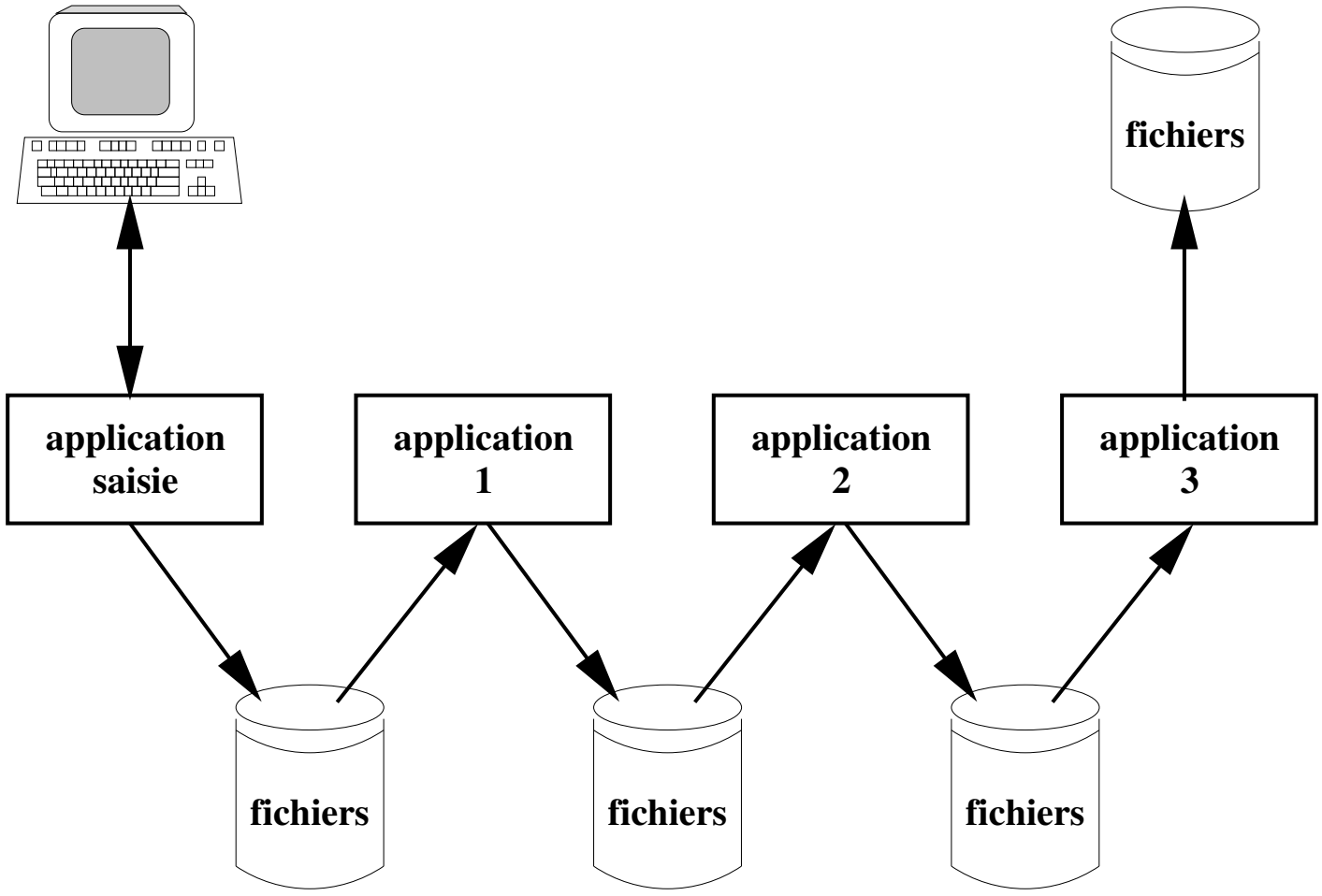
Réalisation des codeurs/décodeurs

1 à 3 homme-jours pour 1 langage.

Table des matières

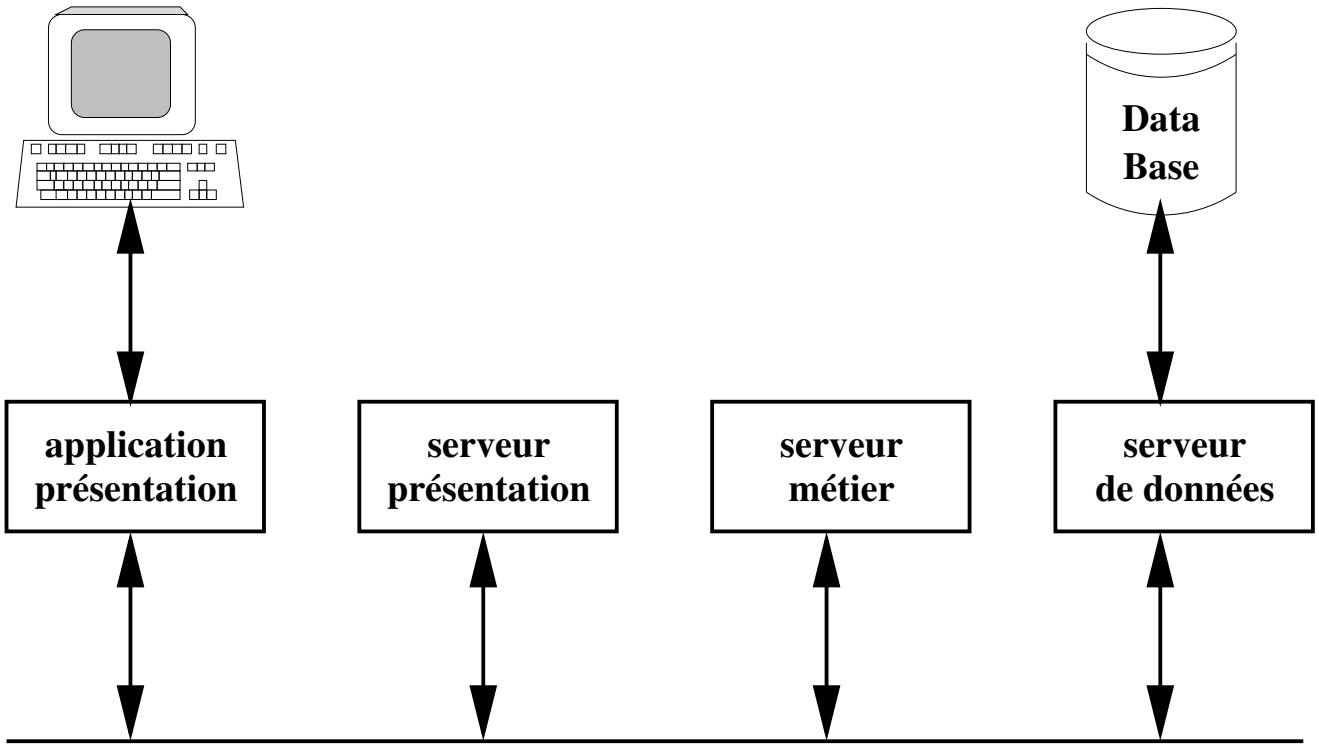
Applications type: Dans le passé

- machine centrale puissante
- applications batch séquentialisées
- communication par fichiers



Applications type: Aujourd'hui

- Applications 3 tiers
  - séparation des différentes tâches
  - applications décentralisées
  - communication par réseau



Applications type: Aujourd'hui

- Applications liées
  - ftp et telnet:
    identification
  - word, exel et gimp:
    copier/coller
  - dans une entreprise: les applications de gestion comptable, du personnel (Ressource Humaine), les outils de travail, . . . partagent les informations (nom, mail, grade, localisation, . . .)

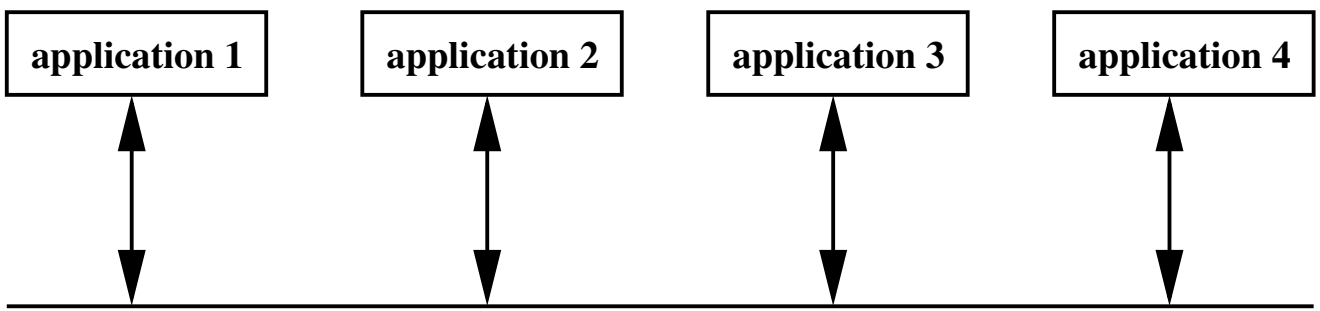


Table des matières

Outils réseau

développer de telles applications

besoin d'outils masquant l'utilisation d'un réseau

Générateur de PDUs

Middleware

Annuaire

développer de telles applications

besoin d'outils masquant l'utilisation d'un réseau

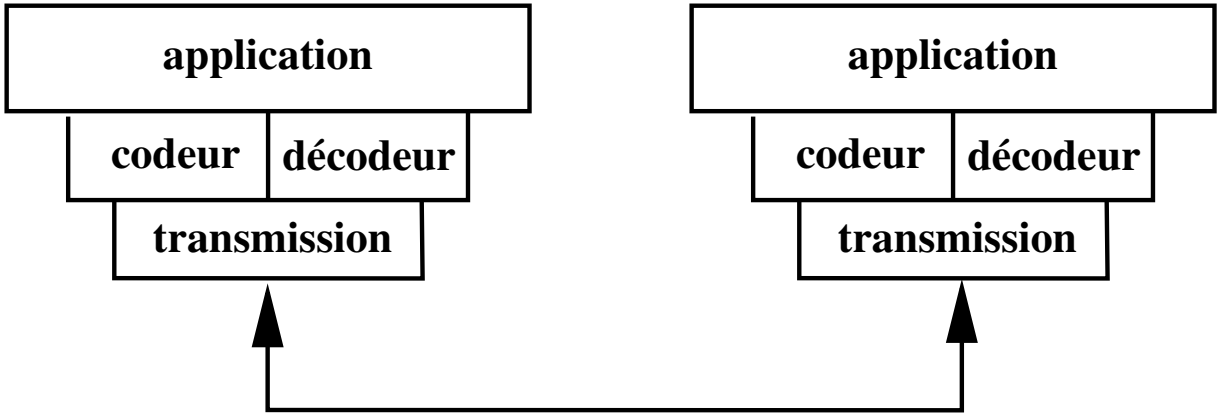
Générateur de PDUs

Middleware

Annuaire

Table des matières

Objectifs

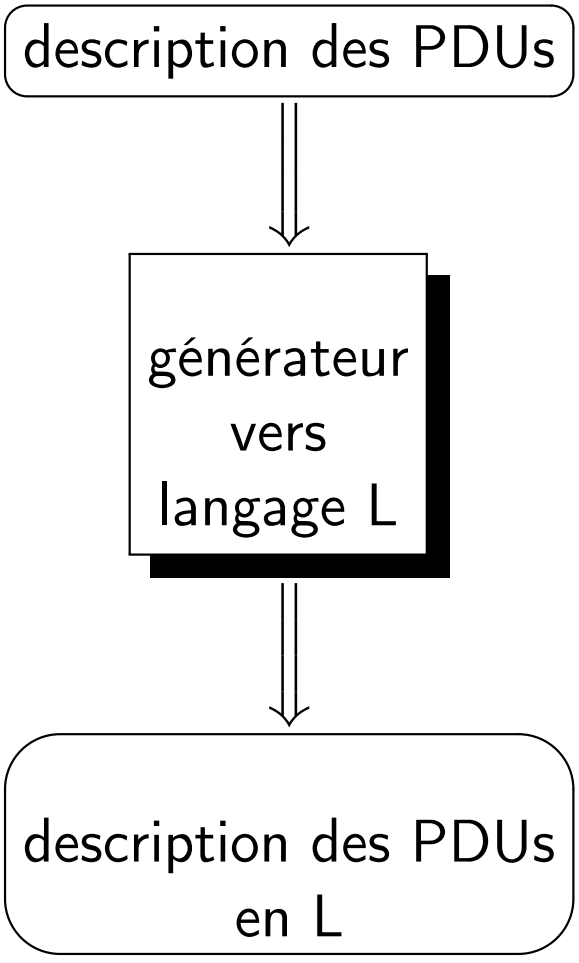


Générer automatiquement les codeurs et les décodeurs

- Ne diminue pas le nombre d'occurrences (**M x L**).
- Facilite l'intéropérabilité des applications entre machines et langages.

Table des matières

Principe


description des PDUs
générateur
vers
langage L
description des PDUs
en L

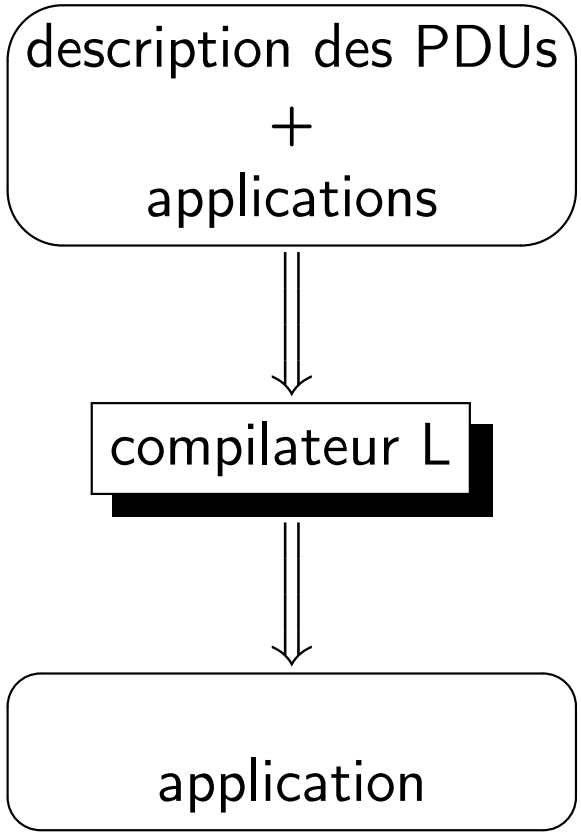

description des PDUs
+
applications
compilateur L
application

Table des matières

XDR: Caractéristiques

- Sun Microsystem
- Fichier source décrit la structure de donnée
  - types de base: enum, u/char, u/int, string, opaque ...
  - types composés: tableau, liste, structure, union
- Utilisé dans NFS, NIS.

XDR: Mini-serveur financier

les indices: CAC40, DOW JONES, NIKKEI, ...

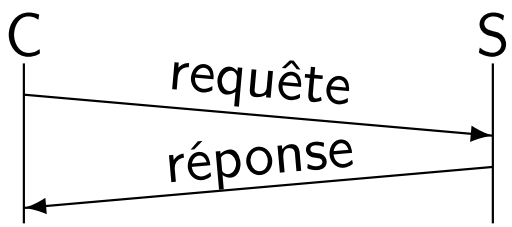
les services:

1) obtention de la valeur du jour d'un indice.
2) obtention des valeurs des 10 derniers jours d'un indice.

modèle réseau:

protocole:

question/réponse



services 1

- PDU_0 requête: getcur, indice
- PDU_1 reponse: status, réel

services 2

- DU_2 requête: getlast, indice
- DU_3 reponse: status, nb, $\{réel_0, réel_1, ...\}$

XDR: Fichiers source & générés

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER  = 1
};
```

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER = 1,
};
typedef enum Tmesstype Tmesstype;
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
typedef enum Treqtype Treqtype;
```

```
struct Trequest {
    Treqtype type;
    char     indice[10];
};
```

```
struct Trequest{
    Treqtype type;
    char     indice[10];
};
typedef struct Trequest Trequest;
```

XDR: Fichiers source & générés

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER  = 1
};
```

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER = 1,
};
typedef enum Tmesstype Tmesstype;
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
typedef enum Treqtype Treqtype;
```

```
struct Trequest {
    Treqtype type;
    char     indice[10];
};
```

```
struct Trequest{
  Treqtype type;
  char     indice[10];
};
typedef struct Trequest Trequest;
```

XDR: Fichiers source & générés

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER  = 1
};
```

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER = 1,
};
typedef enum Tmesstype Tmesstype;
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
typedef enum Treqtype Treqtype;
```

```
struct Trequest {
    Treqtype type;
    char     indice[10];
};
```

```
struct Trequest{
  Treqtype type;
  char     indice[10];
};
typedef struct Trequest Trequest;
```

XDR: Fichiers source & générés

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER  = 1
};
```

```
enum Tmesstype {
     REQUEST = 0,
     ANSWER = 1,
};
typedef enum Tmesstype Tmesstype;
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
typedef enum Treqtype Treqtype;
```

```
struct Tanswer {
  uint    status;
  float  value<10>;
};
```

```
struct Tanswer {
  uint status;
  struct {
     u_int value_len;
     float *value_val;
  } value;
};
typedef struct Tanswer Tanswer;
```

XDR: Fichiers source & générés

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER  = 1
};
```

```
enum Tmesstype {
    REQUEST = 0,
    ANSWER = 1,
};
typedef enum Tmesstype Tmesstype;
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
```

```
enum Treqtype {
    GetCourant  = 0,
    GetPasse    = 1
};
typedef enum Treqtype Treqtype;
```

```
struct Tanswer {
  uint    status;
  float  value<10>;
};
```

```
struct Tanswer {
  uint status;
  struct {
     u_int value_len;
     float *value_val;
  } value;
};
typedef struct Tanswer Tanswer;
```

XDR: Fichiers source & générés

```
union Tmessage  {
  uint  ident;
  switch (Tmesstype kind) {
    case REQUEST:
      Trequest    req;
    case ANSWER:
      Tanswer     ans;
  }
};
```

```
struct Tmessage {
    uint ident;
    Tmesstype kind;
    union {
        Trequest  req;
        Tanswer   ans;
    } u;
}
typedef struct Tmessage Tmessage
```

Interface générée

```
extern  bool_t xdr_Tmesstype (XDR *, Tmesstype*);
extern  bool_t xdr_Treqtype (XDR *, Treqtype*);
extern  bool_t xdr_Trequest (XDR *, Trequest*);
extern  bool_t xdr_Tanswer (XDR *, Tanswer*);
extern  bool_t xdr_Tmessage (XDR *, Tmessage*);
```

XDR: Fichiers source & générés

```
union Tmessage  {
  uint  ident;
  switch (Tmesstype kind) {
    case REQUEST:
      Trequest    req;
    case ANSWER:
      Tanswer     ans;
  }
};
```

```
struct Tmessage {
    uint ident;
    Tmesstype kind;
    union {
        Trequest  req;
        Tanswer   ans;
    } u;
}
typedef struct Tmessage Tmessage
```

Interface générée

```
extern  bool_t xdr_Tmesstype (XDR *, Tmesstype*);
extern  bool_t xdr_Treqtype (XDR *, Treqtype*);
extern  bool_t xdr_Trequest (XDR *, Trequest*);
extern  bool_t xdr_Tanswer (XDR *, Tanswer*);
extern  bool_t xdr_Tmessage (XDR *, Tmessage*);
```

XDR: Encodage

Tout est aligné sur 32 bits

requête CAC40

0	00	00	10	00	ident =4096
4	00	00	00	00	kind=REQUEST
8	00	00	00	00	kind=GetCurr
C	C	A	C	4	
10	0	00	00	00	
14	00	00			

réponse avec 2 floats

0	00	00	10	00	ident =4096
4	00	00	00	01	kind=ANSWER
8	00	00	00	00	status=0
C	00	00	00	02	2
10	01	10	00	0A	float 0
14	01	22	01	EE	float 1

XDR: Encodage

Tout est aligné sur 32 bits

requête CAC40

0	00	00	10	00	ident =4096
4	00	00	00	00	kind=REQUEST
8	00	00	00	00	kind=GetCurr
C	C	A	C	4	
10	0	00	00	00	
14	00	00			

réponse avec 2 floats

0	00	00	10	00	ident =4096
4	00	00	00	01	kind=ANSWER
8	00	00	00	00	status=0
C	00	00	00	02	2
10	01	10	00	0A	float 0
14	01	22	01	EE	float 1

Caractéristiques

- issue de OSI
- fichier source décrit la structure de donnée
  - types de base: BOOLEAN, INTEGER, REAL, bit string, octet string, ...
  - types normalisés: PrintableString, NumericString, date, ...
  - types composés: tableau, liste, structure, union, ensemble
  - possibilité de définir un gabarit de message (généricité)
  - gestion des versions
    - définition des messages (version N)
    - $APP1(N) \longrightarrow APP2$ (N)
    - redéfinition des messages (ajout de champs) (version N+1)
    - mise à jour de APP1 • $APP1(N+1) \longrightarrow APP2$ (N)
- Utilisé dans SNMP.

Fichier source

```
mesPDUs DEFINITION EXPLICIT TAGS := BEGIN
  TrequestData ::= SEQUENCE {
    type     ENUMERATED{GetCourant,GetPasse},
    indice   PrintableString( SIZE(1..10)),
  }
  TanswerData ::= SEQUENCE {
    status    INTEGER,
    value     INTEGER (SIZE(1..10))
  }

  ...

END
```

Codage

plusieurs formats d'encodage

- BER: Basic Encoding Rules
- PER: Packed Encoding Rules
- XML:

Encodege BER

- tout type est codé par: "tag taille valeur"
- le tag est un identifiant de type. Pour les types de base, le tag est prédéfini, pour les autres il est soit généré, soit fixé par l'utilisateur.

tag $<$ 128	1 octet	1xxxxxxx
tag $<$ 2**14	2 octets	0xxxxxxx 1xxxxxxx
tag $<$ 2**21	3 octets	0xxxxxxx 0xxxxxxx 1xxxxxxx
...	...	...

- la taille est codée comme le tag.
- contient la valeur si type de base ou constante, d'autres types "tag taille valeur" si type composé.

XML: Caractéristiques

- De plus en plus utilisé
- Définition des PDU: une dtd
- Emission d'un PDU:
  1. construire l'arbre XML du PDU à partir de sa SD,
  2. le sauver en XML,
  3. l'envoyer.
- Réception d'un PDU:
  1. lire une description XML,
  2. construire l'arbre XML,
  3. remplir sa SD à partir de l'arbre XML.
- Outils: Bibliothèques XML pour construruire des arbres XML (envoie) et rechercher des données (réception).

XML: DTD

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT pdu (request | answer)>
  <!ELEMENT request EMPTY>
    <!ATTLIST request type (courant|passe) #REQUIRED>
    <!ATTLIST request indice (CAC40|NIKKEI|...) #REQUIRED>
  <!ELEMENT answer (value*)>
    <!ATTLIST answer status (OK|ERR) #REQUIRED>
    <!ELEMENT value (#PCDATA)>
```

XML: Encodage

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT pdu (request | answer)>
  <!ELEMENT request EMPTY>
    <!ATTLIST request type (courant|passe) #REQUIRED>
    <!ATTLIST request indice (CAC40|NIKKEI|...) #REQUIRED>
  <!ELEMENT answer (value*)>
    <!ATTLIST answer status (OK|ERR) #REQUIRED>
    <!ELEMENT value (#PCDATA)>
```

```
<?xml version="1.0"?>
<!DOCTYPE pdu SYSTEM "pdu.dtd">
<request type="courant" indice="NIKKEI"/>
```

```
<?xml version="1.0"?>
<!DOCTYPE pdu SYSTEM "pdu.dtd">
<request type="passe" indice="NIKKEI"/>
```

XML: Encodage

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT pdu (request | answer)>
  <!ELEMENT request EMPTY>
    <!ATTLIST request type (courant|passe) #REQUIRED>
    <!ATTLIST request indice (CAC40|NIKKEI|...) #REQUIRED>
  <!ELEMENT answer (value*)>
    <!ATTLIST answer status (OK|ERR) #REQUIRED>
    <!ELEMENT value (#PCDATA)>
```

```
<?xml version="1.0"?>
<!DOCTYPE pdu SYSTEM "pdu.dtd">
<answer status="ERR"/>
```

XML: Encodage

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!ELEMENT pdu (request | answer)>
  <!ELEMENT request EMPTY>
    <!ATTLIST request type (courant|passe) #REQUIRED>
    <!ATTLIST request indice (CAC40|NIKKEI|...) #REQUIRED>
  <!ELEMENT answer (value*)>
    <!ATTLIST answer status (OK|ERR) #REQUIRED>
    <!ELEMENT value (#PCDATA)>
```

```
<?xml version="1.0"?>
<!DOCTYPE pdu SYSTEM "pdu.dtd">
<answer status="OK">
 <value>2.3</value>
 <value>2.5</value>
 <value>2.4</value>
</answer>
```

Table des matières

Comparaison

- Puissance de description
- Efficacité du codage
- Coût CPU du codage/décodage
- Contrôle d'erreurs de programmation
- Facilité d'utilisation

Comparaison

- Puissance de description ASN.1 > XDR > XML
- Efficacité du codage
- Coût CPU du codage/décodage
- Contrôle d'erreurs de programmation
- Facilité d'utilisation

Comparaison

- Puissance de description ASN.1 > XDR > XML
- Efficacité du codage ASN.1 > XDR > XML
- Coût CPU du codage/décodage
- Contrôle d'erreurs de programmation
- Facilité d'utilisation

Comparaison

- Puissance de description ASN.1 > XDR > XML
- Efficacité du codage ASN.1 > XDR > XML
- Coût CPU du codage/décodage XML > ASN.1 > XDR
- Contrôle d'erreurs de programmation

  XDR & ASN.1 : les générateurs font tous les contrôles.

  XML les librairies XML ne vérifient que les structures XML:

```
<?xml version="1.0"?>
<!DOCTYPE pdu SYSTEM "pdu.dtd">
<answer status="OK">
 <value>2.3</value>
 <value>2.5</value>
 <value>2.4</value>
</answer>
```

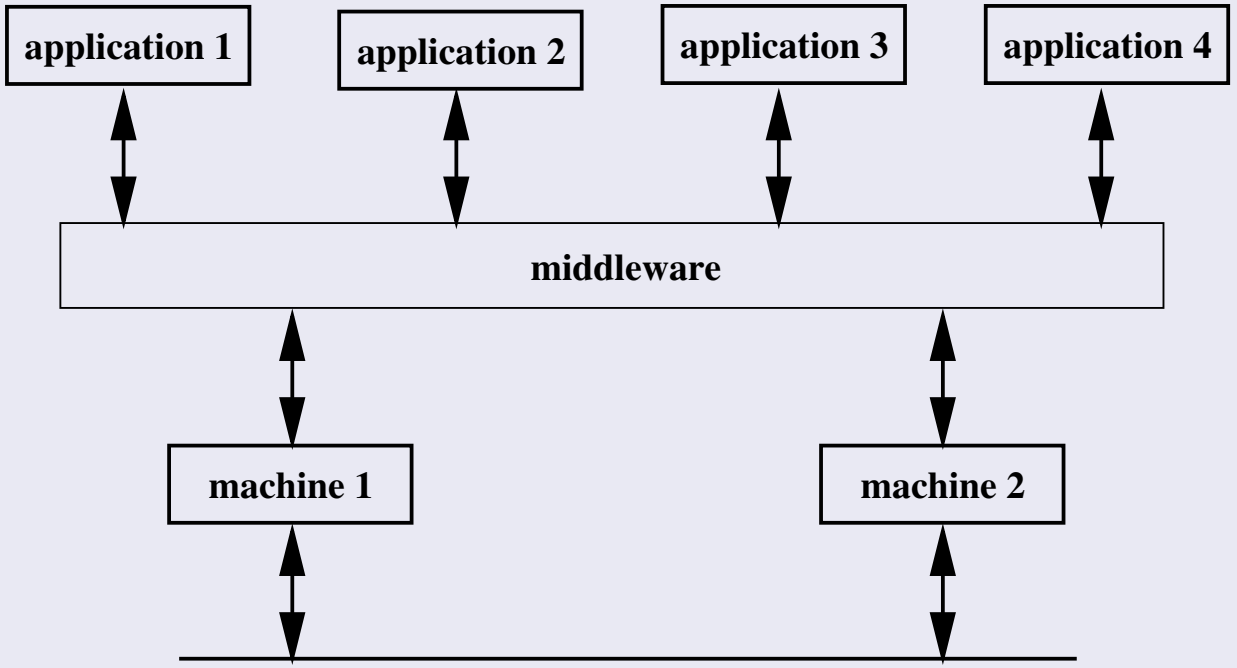
- Facilité d'utilisation

Comparaison

- Puissance de description ASN.1 > XDR > XML
- Efficacité du codage ASN.1 > XDR > XML
- Coût CPU du codage/décodage XML > ASN.1 > XDR
- Contrôle d'erreurs de programmation XML > ASN.1 > XDR
- Facilité d'utilisation

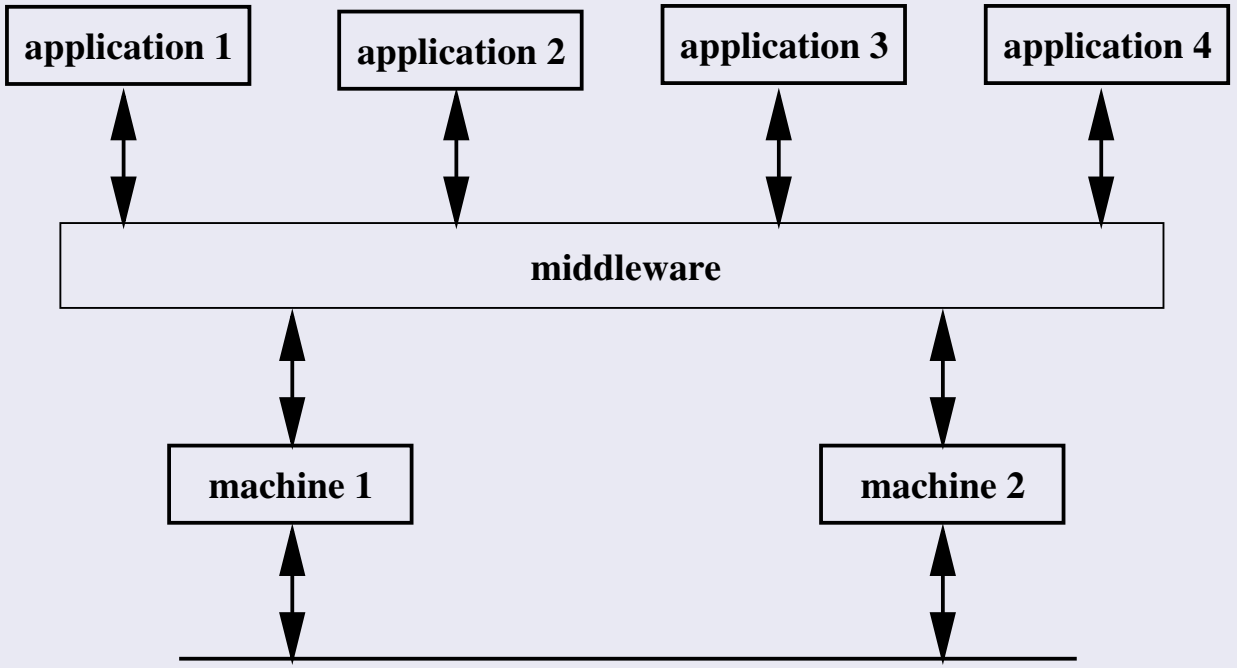
Table des matières

Concept



- abstraction des PDUs
- abstraction des supports de communication
- abstraction de la location physique
- abstraction des langages

Concept



$\Longrightarrow$ Conception plus rapide

$\Longrightarrow$ Interopérabilité des systèmes

$\Longrightarrow$ Interopérabilité des langages

Table des matières

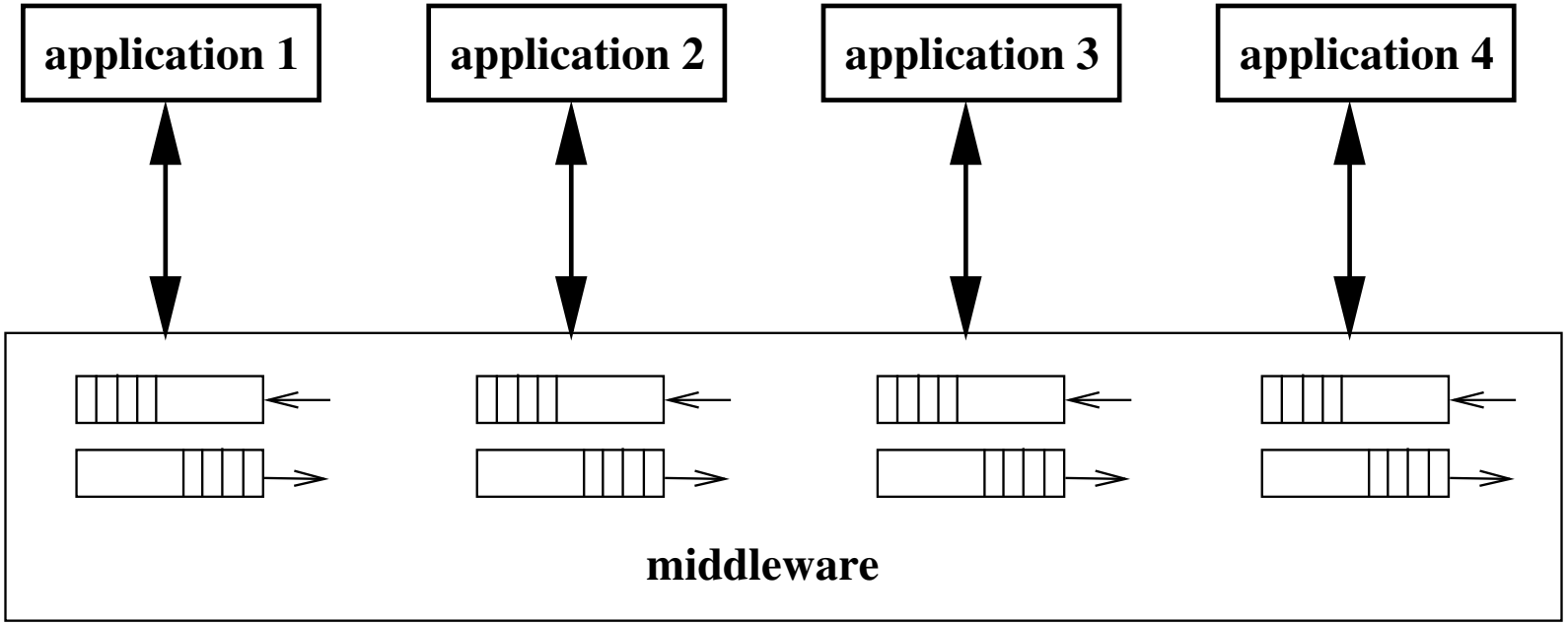
Types

Basés sur messages et événements

Basés sur appels de procdures (RPC, RMI)

Basés sur objet (CORBA, ISE, DCOM, DOTNET)

Basés sur messages et événements

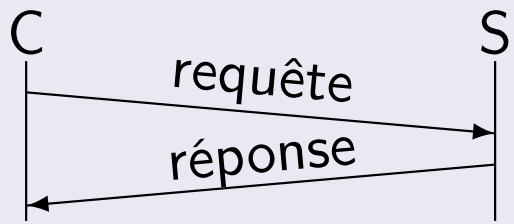


- les serveurs déclarent leurs services
- les clients s'abonnent aux services
- le middleware gére la diffusion au travers de queues
- le middleware est permanent

Basés sur appels de procédures

Principe

protocole: question(client)/réponse(serveur)



coté client: une question est faite par un appel de fonction

coté serveur: le serveur exécute la fonction.

Outil:

- Outil génère la fonction à 100% coté client (souche cliente),
- Outil génére le squelette du serveur

limitations:

1. **Error handling:**

   failures of the remote server or network must be handled when using remote procedure calls.

2. **Parameters, Global variables and side-effects:**

   since the server does not have access to the client's address space, hidden arguments cannot be passed as global variables or returned as side effects.

3. **Performance:**

   remote procedures usually operate one or more orders of magnitude slower than local procedure calls.

4. **Authentication:**

   since remote procedure calls can be transported over unsecured networks, authentication may be necessary. Authentication prevents one entity from masquerading as some other entity.

Basés sur objet

Une extension des RPC où les clients manipulent des objets qui tournent sur des serveurs (voir section 64).

Table des matières

ONC RPC: Open Network Computing: Eléments

Langage d'entrée

```
struct Tanswer { ... }
struct Trequest { ... }
program Exemple {
  version ExVers {
    float fonction_1(int r) = 1;
    Tanswer fonction_2(Trequest r) = 2;
  } = 3;
} = 0x20002000;
```

Identifiant RPC

0-1fffffff	defined by rpc@sun.com
20000000-3fffffff	defined by user
40000000-5fffffff	transient (serveur dynamique)
60000000- ffffffff	reserved

Serveur d'enregistrement

ONC RPC: Open Network Computing: Eléments

Langage d'entrée

```
struct Tanswer { ... }
struct Trequest { ... }
program Exemple {
  version ExVers {
    float fonction_1(int r) = 1;
    Tanswer fonction_2(Trequest r) = 2;
  } = 3;
} = 0x20002000;
```

Identifiant RPC

0-1fffffff	defined by rpc@sun.com
20000000-3fffffff	defined by user
40000000-5fffffff	transient (serveur dynamique)
60000000- ffffffff	reserved

Serveur d'enregistrement

ONC RPC: Open Network Computing: Eléments

Langage d'entrée

```
struct Tanswer { ... }
struct Trequest { ... }
program Exemple {
  version ExVers {
    float fonction_1(int r) = 1;
    Tanswer fonction_2(Trequest r) = 2;
  } = 3;
} = 0x20002000;
```

Identifiant RPC

0-1fffffff	defined by rpc@sun.com
20000000-3fffffff	defined by user
40000000-5fffffff	transient (serveur dynamique)
60000000- ffffffff	reserved

Serveur d'enregistrement

ONC RPC: Open Network Computing: Eléments

Langage d'entrée

```
struct Tanswer { ... }
struct Trequest { ... }
program Exemple {
  version ExVers {
    float fonction_1(int r) = 1;
    Tanswer fonction_2(Trequest r) = 2;
  } = 3;
} = 0x20002000;
```

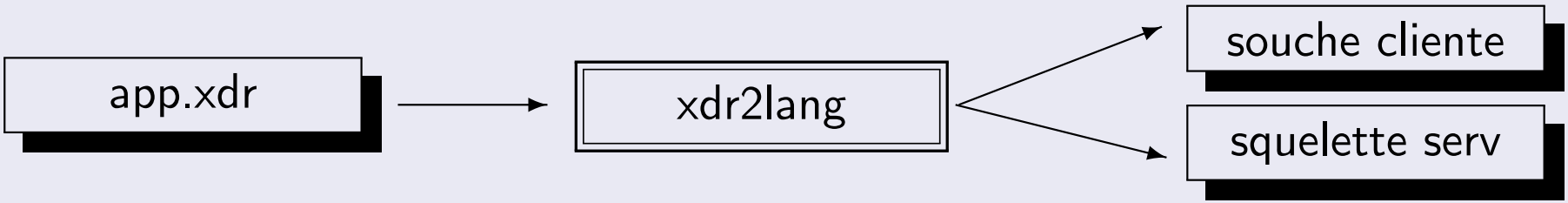
Identifiant RPC

0-1fffffff	defined by rpc@sun.com
20000000-3fffffff	defined by user
40000000-5fffffff	transient (serveur dynamique)
60000000- ffffffff	reserved

Serveur d'enregistrement

ONC RPC: Open Network Computing: Eléments

Un générateur de code



Génération d'un serveur

- Ecrire le code des fonctions RPC.
- Le compiler avec le squelette du serveur.

Génération d'un client

- Ecrire un programme principal en utilsant les fonctions RPC.
- Le compiler avec la souche cliente.

Table des matières

ONC RPC: Open Network Computing: Source XDR & rpcgen

Fichier d'entrée: finance.x

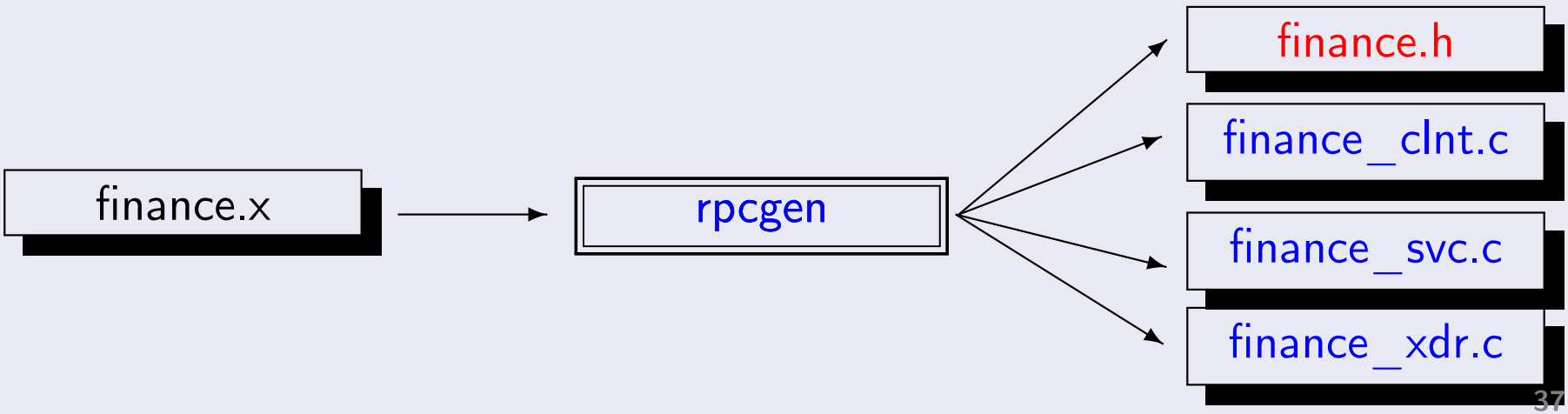
```
struct Tanswer { int status; float value<10>; }
program miniserv {
  version VERS {
    Tanswer getCurr(string indice) = 1;
    Tanswer getLast(string indice) = 2;
  } = 2;
} = 0x20002000 ;
```

ONC RPC: Open Network Computing: Source XDR & rpcgen

Fichier d'entrée: finance.x

```
struct Tanswer { int status; float value<10>; }
program miniserv {
  version VERS {
    Tanswer getCurr(string indice) = 1;
    Tanswer getLast(string indice) = 2;
  } = 2;
} = 0x20002000 ;
```

rpcgen



ONC RPC: Open Network Computing: Source XDR & rpcgen

Fichier d'entrée: finance.x

```
struct Tanswer { int status; float value<10>; }
program miniserv {
  version VERS {
    Tanswer getCurr(string indice) = 1;
    Tanswer getLast(string indice) = 2;
  } = 2;
} = 0x20002000 ;
```

finance.h

```
struct Tanswer {
  int status;
  struct {
    u_int value_len;
    float *value_val;
  } value;
};
extern  bool_t xdr_Tanswer (XDR *, Tanswer*);
```

ONC RPC: Open Network Computing: Source XDR & rpcgen

Fichier d'entrée: finance.x

```
struct Tanswer { int status; float value<10>; }
program miniserv {
  version VERS {
    Tanswer getCurr(string indice) = 1;
    Tanswer getLast(string indice) = 2;
  } = 2;
} = 0x20002000 ;
```

finance.h

```
...
extern Tanswer* getCurr_2(char**, CLIENT*);
extern Tanswer* getLast_2(char**, CLIENT*);
extern Tanswer* getCurr_2_svc(char**, struct svc_req*);
extern Tanswer* getLast_2_svc(char**, struct svc_req*);
```

ONC RPC: Open Network Computing: Implantation d'un serveur

"finance_svc.c" contient le squelette du serveur complet. Ce qui manque c'est ce que font les fonctions RPC (getcurr_2_svc et getlast_2_svc). Il suffit de les implanter dans un fichier.

Description du serveur (serveur.c)

```
#include "finance.h"

Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
}

Tanswer* getlast_2_svc(char **indice, struct svc_req *cli)
{
}
```

Génération & lancement

```
⟶ cc -o serveur finance_svc.c serveur.c
⟶ ./serveur
```

ONC RPC: Open Network Computing: Implantation d'un serveur

"finance_svc.c" contient le squelette du serveur complet. Ce qui manque c'est ce que font les fonctions RPC (getcurr_2_svc et getlast_2_svc). Il suffit de les implanter dans un fichier.

Description du serveur (serveur.c)

```
#include "finance.h"
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
}
Tanswer* getlast_2_svc(char **indice, struct svc_req *cli)
{
}
```

Génération & lancement

```
⟶ cc -o serveur finance_svc.c serveur.c
⟶ ./serveur
```

ONC RPC: Open Network Computing: Implantation d'un serveur

"finance_svc.c" contient le squelette du serveur complet. Ce qui manque c'est ce que font les fonctions RPC (getcurr_2_svc et getlast_2_svc). Il suffit de les implanter dans un fichier.

Description du serveur (serveur.c)

```
#include "finance.h"

Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
}

Tanswer* getlast_2_svc(char **indice, struct svc_req *cli)
{
}
```

Génération & lancement

```
⟶ cc -o serveur finance_svc.c serveur.c
⟶ ./serveur
```

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{









}
```

ONC RPC: Open Network Computing: Implantation d'un serveur

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
        Tanswer a;









  return &a;
}
```

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
  static Tanswer a;









  return &a;
}
```

ONC RPC: Open Network Computing: Implantation d'un serveur

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
  static Tanswer a;

/** typedef struct Tanswer {                                **/
/**   int status;                                           **/
/**   struct {                                              **/
/**     u_int value_len;                                    **/
/**     float *value_val;                                   **/
/**   } value;                                              **/
/** } Tanswer;                                              **/


  return &a;
}
```

ONC RPC: Open Network Computing: Implantation d'un serveur

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
  static Tanswer a;
  static float    ft[1]; a.value.value_val=ft;
/** typedef struct Tanswer {                                **/
/**   int status;                                           **/
/**   struct {                                              **/
/**     u_int value_len;                                    **/
/**     float *value_val;                                   **/
/**   } value;                                              **/
/** } Tanswer;                                              **/


  return &a;
}
```

ONC RPC: Open Network Computing: Implantation d'un serveur

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
  static Tanswer a;
  static float    ft[1]; a.value.value_val=ft;









  return &a;
}
```

ONC RPC: Open Network Computing: Implantation d'un serveur

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
  static Tanswer a;
  static float    ft[1]; a.value.value_val=ft;
  if ( strcmp(*indice,"CAC40")==0 ) {


  } else if ( strcmp(*indice,"NIKKEI")==0 ) {

  } else {


  }
  return &a;
}
```

ONC RPC: Open Network Computing: Implantation d'un serveur

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
  static Tanswer a;
  static float    ft[1]; a.value.value_val=ft;
  if ( strcmp(*indice,"CAC40")==0 ) {
    a.status=0;
    a.value.value_len=1; a.value.value_val[0] = 3500;
  } else if ( strcmp(*indice,"NIKKEI")==0 ) {
    ...
  } else {



  }
  return &a;
}
```

ONC RPC: Open Network Computing: Implantation d'un serveur

```
Tanswer* getcurr_2_svc(char **indice, struct svc_req *cli)
{
  static Tanswer a;
  static float    ft[1]; a.value.value_val=ft;
  if ( strcmp(*indice,"CAC40")==0 ) {
    a.status=0;
    a.value.value_len=1; a.value.value_val[0] = 3500;
  } else if ( strcmp(*indice,"NIKKEI")==0 ) {
    ...
  } else {
    a.status=-1;
    a.value.value_len=0; a.value.value_val = 0;
  }
  return &a;
}
```

ONC RPC: Open Network Computing: Implantation d'un client

"`finance_clnt.c`" contient les fonctions RPC (`getcurr_2` et `getlast_2`).On peut les utiliser à volonté.

Description du client (client.c)

```
int main (int argc, char *argv[])
{
  // initialisation du RPC
  CLIENT* serv=clnt_create (SERVEUR,miniserv,VERS,"udp");

  // appel RPC
  Tanswer* a=getcurr_2(&argv[2], serv);

  // traitement du résultat de l'appel
  ...

  return 0;
}
```

Génération & lancement

```
⟶ cc -o client finance_clnt.c client.c
⟶ ./client host CAC40
```

ONC RPC: Open Network Computing: Implantation d'un client

"finance_clnt.c" contient les fonctions RPC (getcurr_2 et getlast_2).On peut les utiliser à volonté.

Description du client (client.c)

```
int main (int argc, char *argv[])
{
  // initialisation du RPC
  CLIENT* serv=clnt_create (SERVEUR,miniserv,VERS,"udp");
  // appel RPC
  Tanswer* a=getcurr_2(&argv[2], serv);
  // traitement du résultat de l'appel
  ...
  return 0;
}
```

Génération & lancement

```
⟶ cc -o client finance_clnt.c client.c
⟶ ./client host CAC40
```

ONC RPC: Open Network Computing: Implantation d'un client

"finance_clnt.c" contient les fonctions RPC (getcurr_2 et getlast_2).On peut les utiliser à volonté.

Description du client (client.c)

```
int main (int argc, char *argv[])
{
  // initialisation du RPC
  CLIENT* serv=clnt_create (SERVEUR,miniserv,VERS,"udp");
  // appel RPC
  Tanswer* a=getcurr_2(&argv[2], serv);
  // traitement du résultat de l'appel
  ...
  return 0;
}
```

Génération & lancement

```
⟶ cc -o client finance_clnt.c client.c
⟶ ./client host CAC40
```

ONC RPC: Open Network Computing: Implantation d'un client

"finance_clnt.c" contient les fonctions RPC (getcurr_2 et getlast_2).On peut les utiliser à volonté.

Description du client (client.c)

```
int main (int argc, char *argv[])
{
  // initialisation du RPC
  CLIENT* serv=clnt_create (SERVEUR,miniserv,VERS,"udp");
  // appel RPC
  Tanswer* a=getcurr_2(&argv[2], serv);
  // traitement du résultat de l'appel
  ...
  return 0;
}
```

Génération & lancement

```
⟶ cc -o client finance_clnt.c client.c
⟶ ./client host CAC40
```

ONC RPC: Open Network Computing: Implantation d'un client

```
int main (int argc, char *argv[])
{
  // initialisation du RPC

  // appel RPC

  // traitement du résultat de l'appel




  return 0;
}
```

ONC RPC: Open Network Computing: Implantation d'un client

```
int main (int argc, char *argv[])
{
  // initialisation du RPC

  // appel RPC

  // traitement du résultat de l'appel




  return 0;
}
```

```
  CLIENT *serv;
  if ( (serv=clnt_create (argv[1],miniserv,VERS,"udp"))==0 ) {
    clnt_pcreateerror (argv[1]);
    exit (1);
```

ONC RPC: Open Network Computing: Implantation d'un client

```
int main (int argc, char *argv[])
{
  // initialisation du RPC
  CLIENT* serv=clnt_create (SERVEUR,miniserv,VERS,"udp");
  // appel RPC

  // traitement du résultat de l'appel




  return 0;
}
```

```
  Tanswer* a;
  if ( (a=getcurr_2(&argv[2], serv))==0 ) {
    clnt_perror (serv, "call failed");
    exit(1);
```

ONC RPC: Open Network Computing: Implantation d'un client

```
int main (int argc, char *argv[])
{
  // initialisation du RPC
  CLIENT* serv=clnt_create (SERVEUR,miniserv,VERS,"udp");
  // appel RPC
  Tanswer* a=getcurr_2(&argv[2], serv);
  // traitement du résultat de l'appel
  if (a->status<0) {
    fprintf(stderr,"indice inconnu: %s\n",argv[2]);
  } else {
    fprintf(stderr,"valeur de %s: %7.2f\n",
        argv[2],a->value.value_val[0]);
  }
  return 0;
}
```

Table des matières

ONC RPC: Open Network Computing: Source XDR & jrpcgen

Fichier d'entrée: finance.x

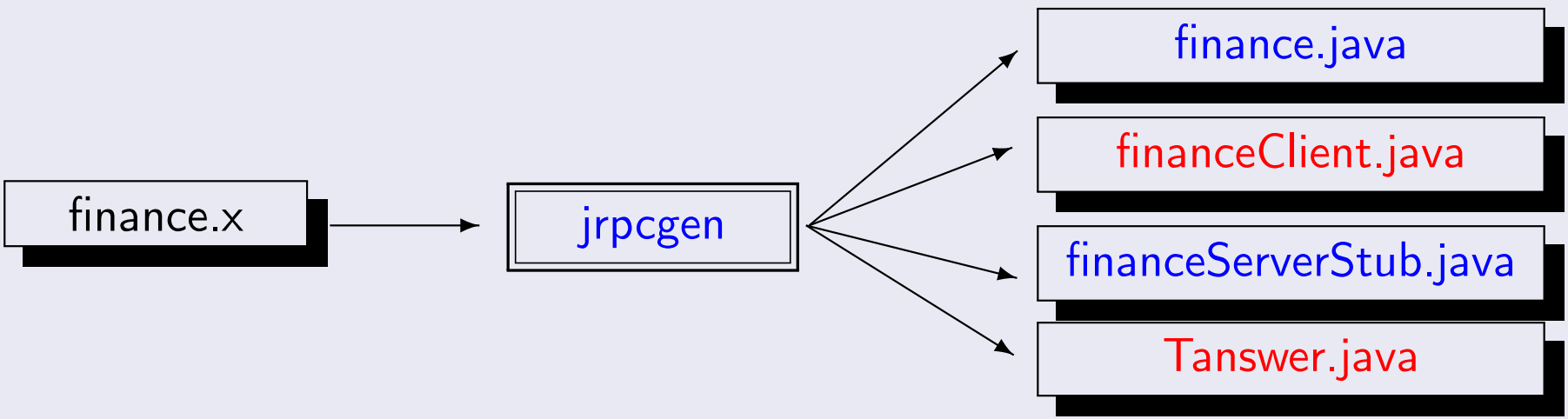
```
struct Tanswer { int status; float value<10>; }
program miniserv {
  version VERS {
    Tanswer getCurr(string indice) = 1;
    Tanswer getLast(string indice) = 2;
  } = 2;
} = 0x20002000 ;
```

ONC RPC: Open Network Computing: Source XDR & jrpcgen

Fichier d'entrée: finance.x

```
struct Tanswer { int status; float value<10>; }
program miniserv {
  version VERS {
    Tanswer getCurr(string indice) = 1;
    Tanswer getLast(string indice) = 2;
  } = 2;
} = 0x20002000 ;
```

jrpcgen



ONC RPC: Open Network Computing: Source XDR & jrpcgen

Fichier d'entrée: finance.x

```
struct Tanswer { int status; float value<10>; }
program miniserv {
  version VERS {
    Tanswer getCurr(string indice) = 1;
    Tanswer getLast(string indice) = 2;
  } = 2;
} = 0x20002000 ;
```

Tanswer.java

```
public class Tanswer implements XdrAble {
    public int status;
    public float [] value;
    public Tanswer() { ... }
    ...
}
```

ONC RPC: Open Network Computing: Source XDR & jrpcgen

Fichier d'entrée: finance.x

```
struct Tanswer { int status; float value<10>; }
program miniserv {
  version VERS {
    Tanswer getCurr(string indice) = 1;
    Tanswer getLast(string indice) = 2;
  } = 2;
} = 0x20002000 ;
```

financeClient.java

```
public class financeClient extends OncRpcClientStub {
    public financeClient(InetAddress host, int protocol)
    public financeClient(InetAddress host, int protocol) {
        super(host, finance.miniserv, 2, 0, protocol);
    }
    public Tanswer getCurr_2(String indice)  { ... }
    public Tanswer getLast_2(String indice) { ... }
    ...
```

ONC RPC: Open Network Computing: Implantation d'un client

La classe "financeClient" contient les membres RPC (getCurr_2 et getLast_2). Dès qu'un objet est créé, on peut les utiliser à volonté.

Description du client (client.java)

```
class client {
 public static void main(String argv[]) {
   // initialisation du RPC
   InetAddress host = InetAddress.getAllByName(argv[0])[0];
   financeClient server = new
       financeClient(host, OncRpcProtocols.ONCRPC_TCP);
   // appel RPC
   Tanswer a=server.getCurr_2(argv[1]);
   // traitement du résultat de l'appel
   if (a.status<0)
   ...
  }};
```

Génération & lancement

```
⟶ javac finance.javac financeClient.java client.java
```

ONC RPC: Open Network Computing: Implantation d'un client

La classe "financeClient" contient les membres RPC (getCurr_2 et getLast_2). Dès qu'un objet est créé, on peut les utiliser à volonté.

Description du client (client.java)

```
class client {
 public static void main(String argv[]) {
   // initialisation du RPC
   InetAddress host = InetAddress.getAllByName(argv[0])[0];
   financeClient server = new
       financeClient(host, OncRpcProtocols.ONCRPC_TCP);
   // appel RPC
   Tanswer a=server.getCurr_2(argv[1]);
   // traitement du résultat de l'appel
   if (a.status<0)
   ...
  }};
```

Génération & lancement

```
⟶ javac finance.javac financeClient.java client.java
```

ONC RPC: Open Network Computing: Implantation d'un client

La classe "financeClient" contient les membres RPC (getCurr_2 et getLast_2). Dès qu'un objet est créé, on peut les utiliser à volonté.

Description du client (client.java)

```
class client {
 public static void main(String argv[]) {
   // initialisation du RPC
   InetAddress host = InetAddress.getAllByName(argv[0])[0];
   financeClient server = new
       financeClient(host, OncRpcProtocols.ONCRPC_TCP);
   // appel RPC
   Tanswer a=server.getCurr_2(argv[1]);
   // traitement du résultat de l'appel
   if (a.status<0)
   ...
  }};
```

Génération & lancement

```
⟶ javac finance.javac financeClient.java client.java
```

ONC RPC: Open Network Computing: Implantation d'un client

La classe "financeClient" contient les membres RPC (getCurr_2 et getLast_2). Dès qu'un objet est créé, on peut les utiliser à volonté.

Description du client (client.java)

```
class client {
 public static void main(String argv[]) {
   // initialisation du RPC
   InetAddress host = InetAddress.getAllByName(argv[0])[0];
   financeClient server = new
       financeClient(host, OncRpcProtocols.ONCRPC_TCP);
   // appel RPC
   Tanswer a=server.getCurr_2(argv[1]);
   // traitement du résultat de l'appel
   if (a.status<0)
   ...
  }};
```

Génération & lancement

```
⟶ javac finance.javac financeClient.java client.java
```

ONC RPC: Open Network Computing: Implantation d'un client

La classe "financeClient" contient les membres RPC (getCurr_2 et getLast_2). Dès qu'un objet est créé, on peut les utiliser à volonté.

Description du client (client.java)

```
class client {
 public static void main(String argv[]) {
   // initialisation du RPC
   InetAddress host = InetAddress.getAllByName(argv[0])[0];
   financeClient server = new
       financeClient(host, OncRpcProtocols.ONCRPC_TCP);
   // appel RPC
   Tanswer a=server.getCurr_2(argv[1]);
   // traitement du résultat de l'appel
   if (a.status<0)
   ...
  }};
```

Génération & lancement

```
⟶ javac finance.javac financeClient.java client.java
```

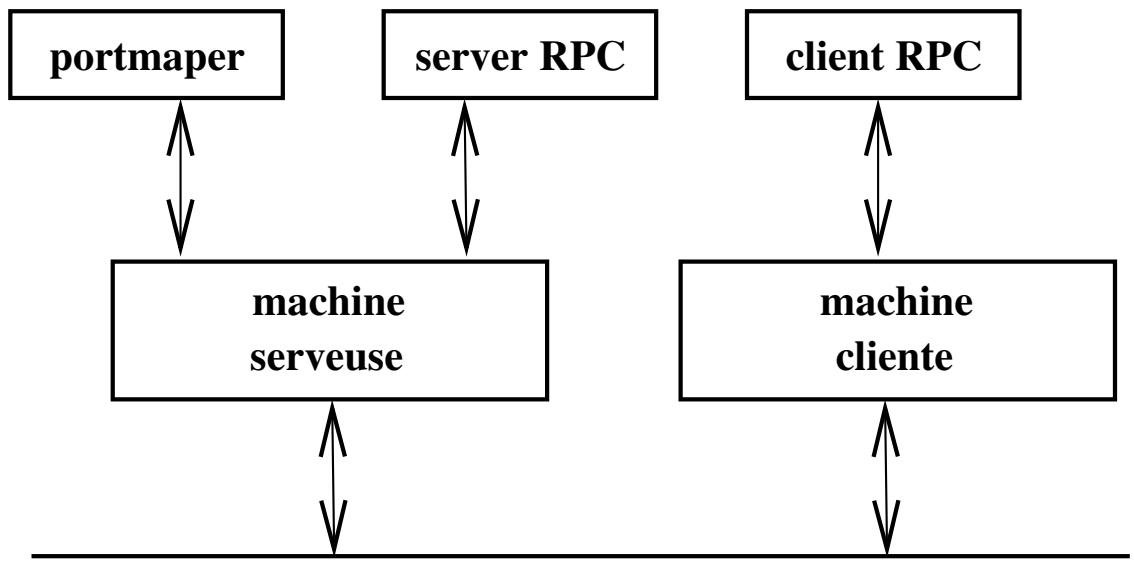
Table des matières

Localisation

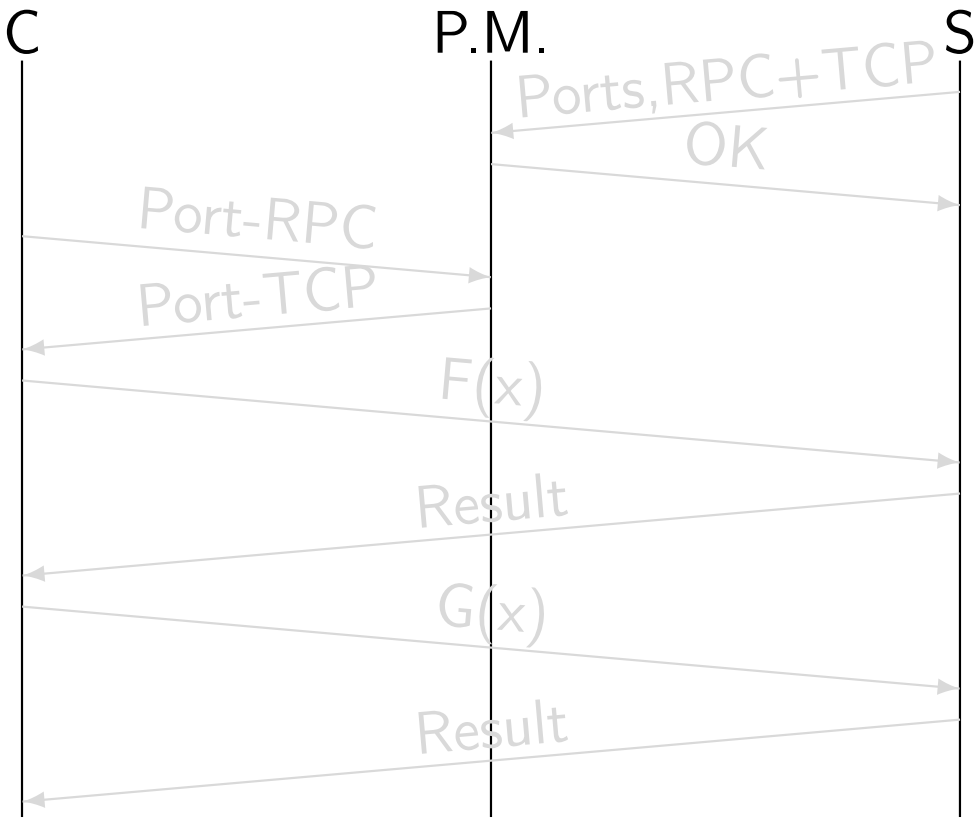
Un client à besoin de connaître:

1. l'addresse réseau de la machine du serveur.
2. l'identifiant du RPC (port RPC).

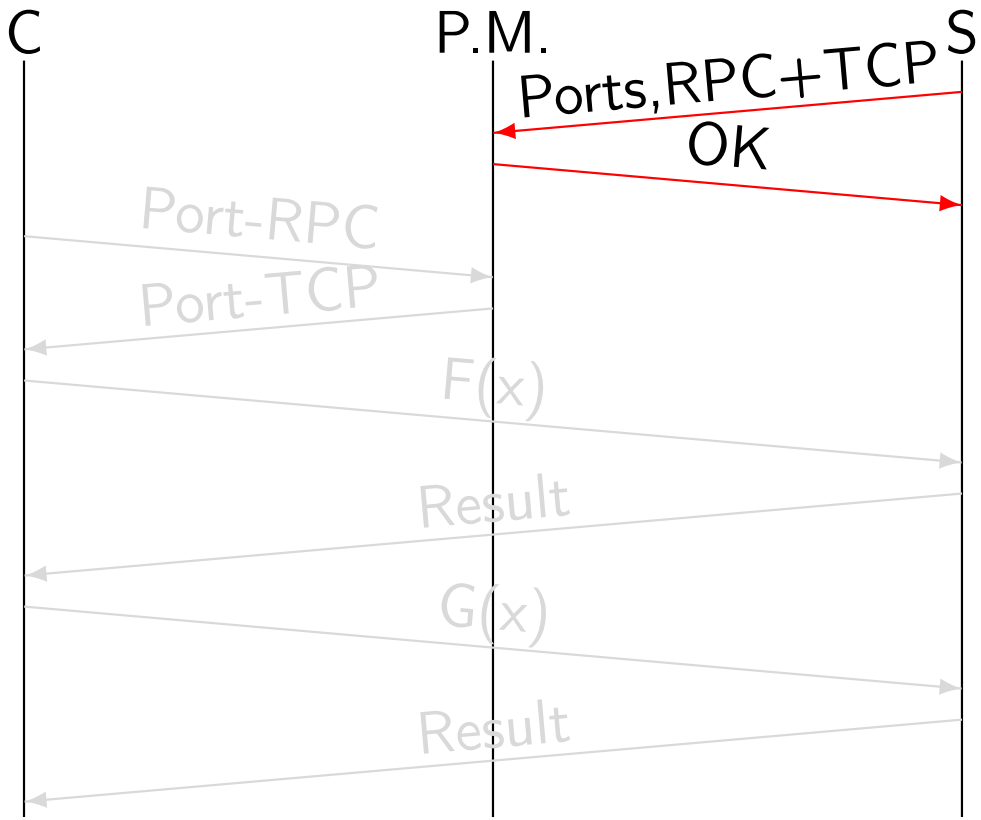
$\Longrightarrow$ Le rôle du port mapper (portmap ou rpcbind sous Unix) est de cross-référencer les ports RPC avec les ports TCP ou IDP. le port mapper et le serveur doivent être sur la même machine.



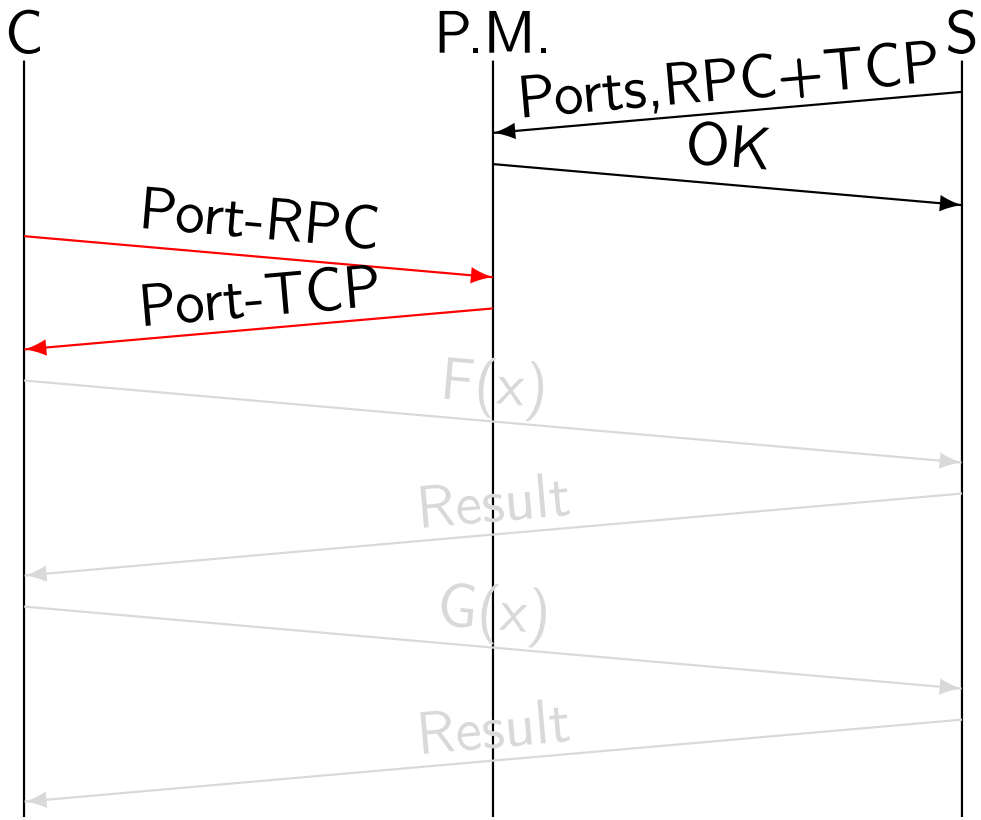
MSC



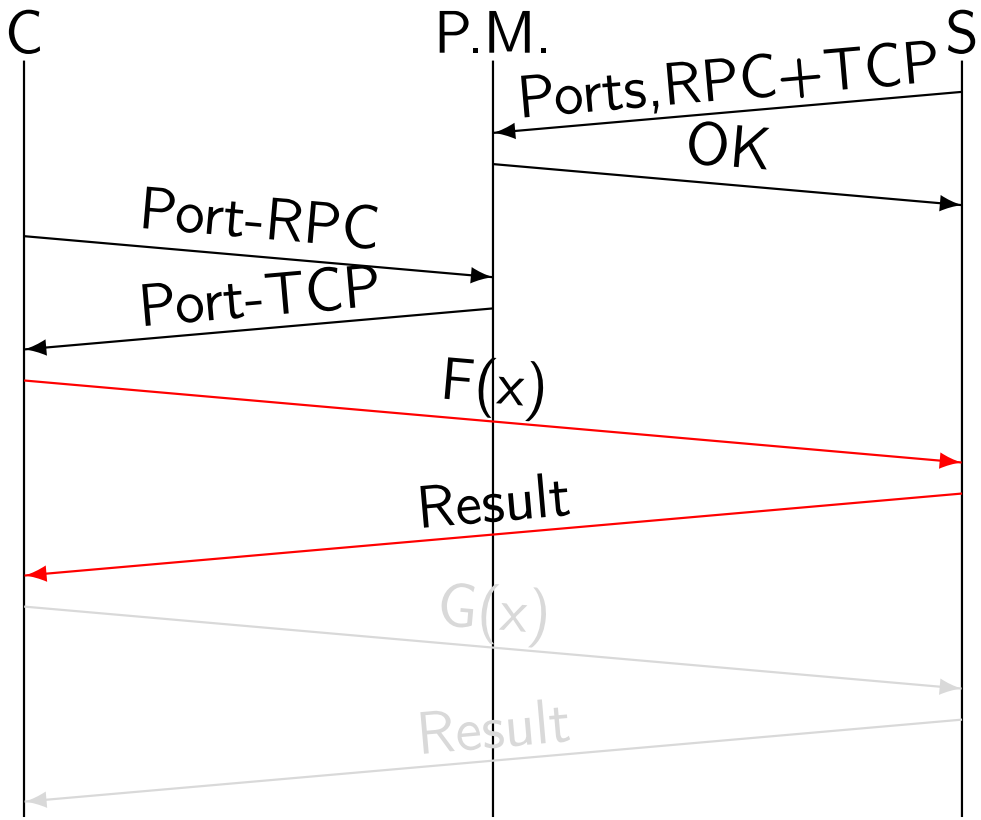
MSC



MSC



MSC



MSC

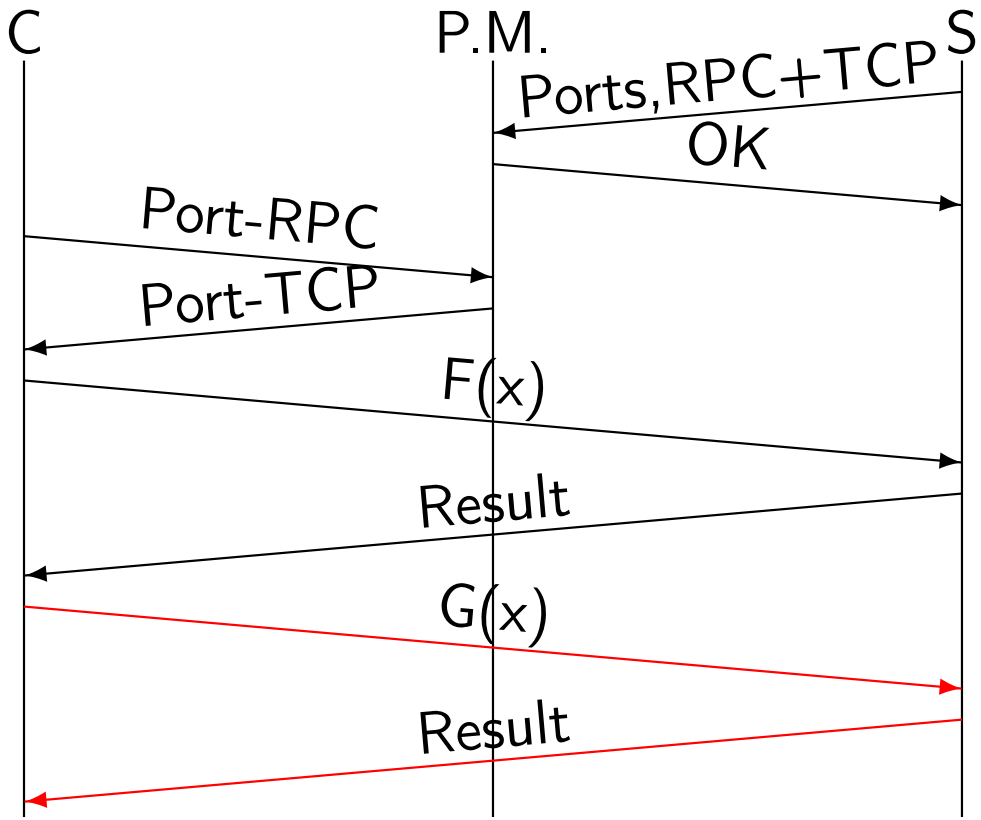


Table des matières

XML RPC-Eléments: Protocole & identifiant

Protocole de transport: HTTP (post pour question)

Identifiant RPC serveur port nom

PDU Ils sont décrits en XML sans DTD.

$\Longrightarrow$ pas de contrôle

XML RPC-Eléments: PDU

Quelques balises XML pour définir:

types de base entier (32 bits), chaîne de caractères, réels, date, ...

```
<i4>41</i4>
<string>bonjour monsieur</string>
```

types composés tableau, structure.

PDU requête nom de la méthode à appeler et ses paramètres

PDU réponse paramètre retourné

PDU d'erreur

XML RPC-Eléments: Introspection

Un serveur XML RPC doit founir au travers de méthodes dont les noms sont prédéfinis les informations suivantes:

1. La liste des fonctions RPC.
2. Pour chaque fonction la structure du paramètre de son PDU requête.
3. Pour chaque fonction la structure du paramètre de son PDU réponse.
4. Pour chaque fonction une documentation.

Table des matières

XML RPC-Principe: Principe

Les outils XML RPC fournissent des bibliothèques dont les services sont définis ci-dessous:

1. Construire un arbre XML.
2. Extraire des valeurs d'un arbre XML.

XML RPC-Principe: Création d'un serveur

- Ecrire les fonctions RPC: Elles ont un argument qui est un arbre XML et elles renvoient un arbre XML.
- Enregistrer les fonctions RPC avec leurs aides et leurs signatures.
- Lire une requête, appeler une fonction qui aiguille sur la bonne fonction ou renvoie un arbre XML erreur.

XML RPC-Principe: Création d'un client

langages interprétés

- Se connecter au serveur $\Longrightarrow$ crée la souche cliente par introspection.
- Construire le XML du paramètre de la requête.
- Appeler la fonction RPC avec ce XML et récupérer le XML de la réponse.
- Extraire du XML renvoyé les valeurs.

langages compilés Très lourd et aucun contrôle.

Table des matières

Langage d'entrée

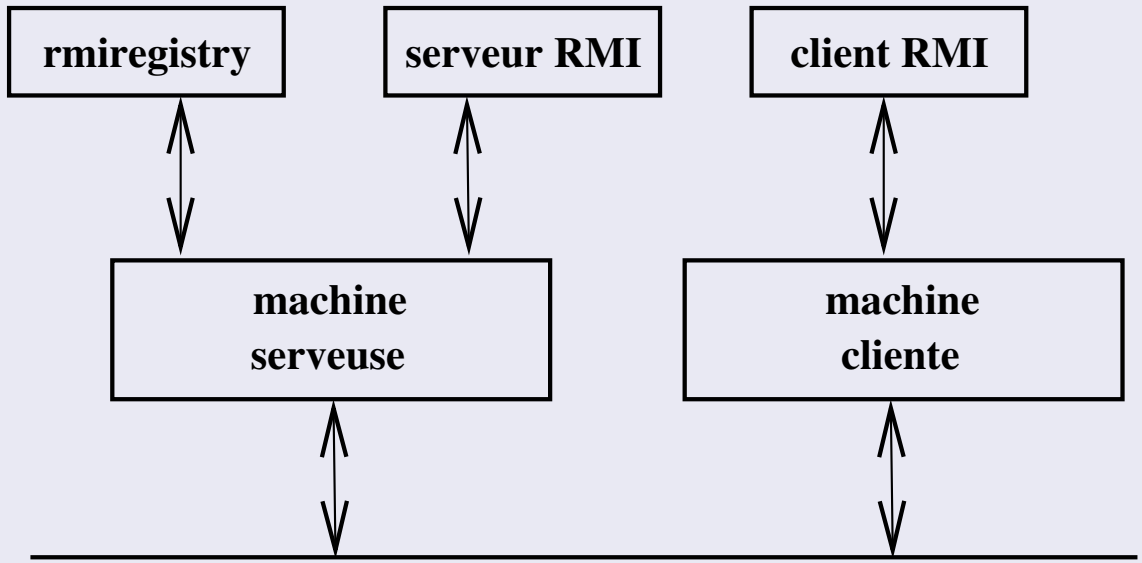
Pas de langage spécifique mais une Interface Java.

```
import java.rmi.*;
public interface exemple extends
java.rmi.Remote {
  public double add(double x, double y)
    throws RemoteException;
  public int log2(int x) throws
RemoteException;
}
```

- Calcule la valeur retournée en fonction des arguments.
- Types des arguments et de la valeur retournée doivent être sérialisables.

Localisation

- Identifiant: une chaîne de caractères
- Rmiregistry: serveur de localisation



⇒ les serveurs s'enregistrent sous un identifiant
⇒ il génère la souches clientes des serveurs
⇒ les clients l'interrogent pour récupérer la souche cliente

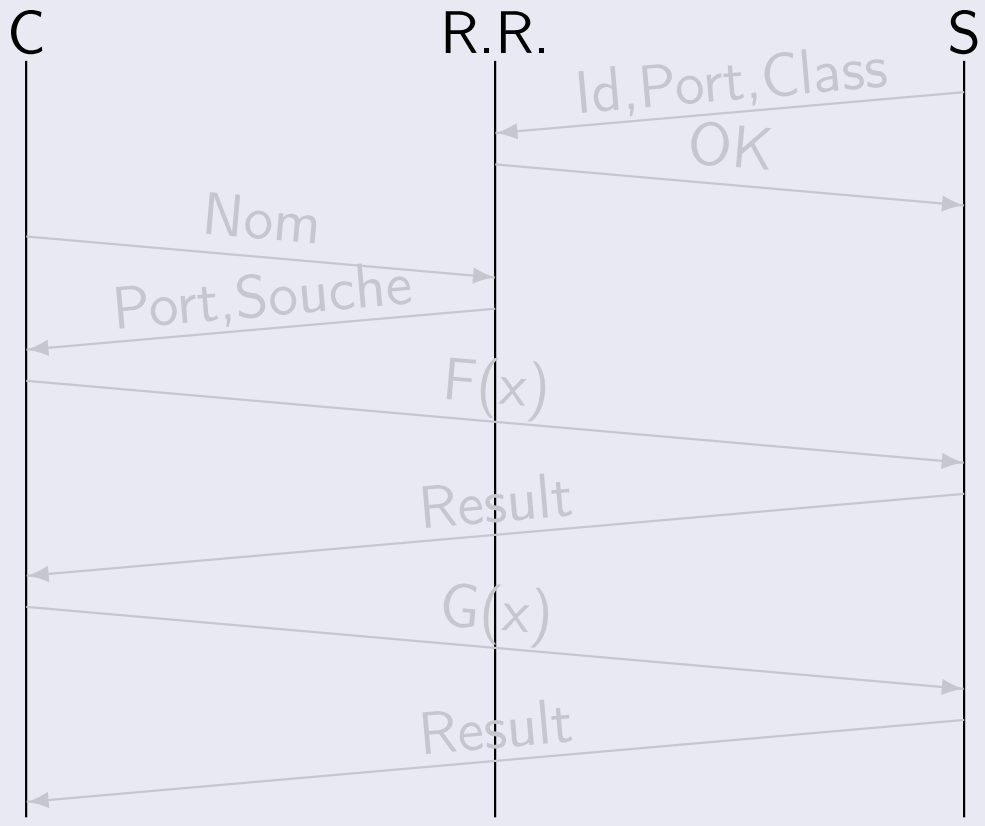
Génération d'un serveur

- Définir une classe implantant l'interface.
- Un main
  - créant une instance de la classe
  - s'enregistrant sur le serveur RMI.
  - entrant dans la boucle d'écoute

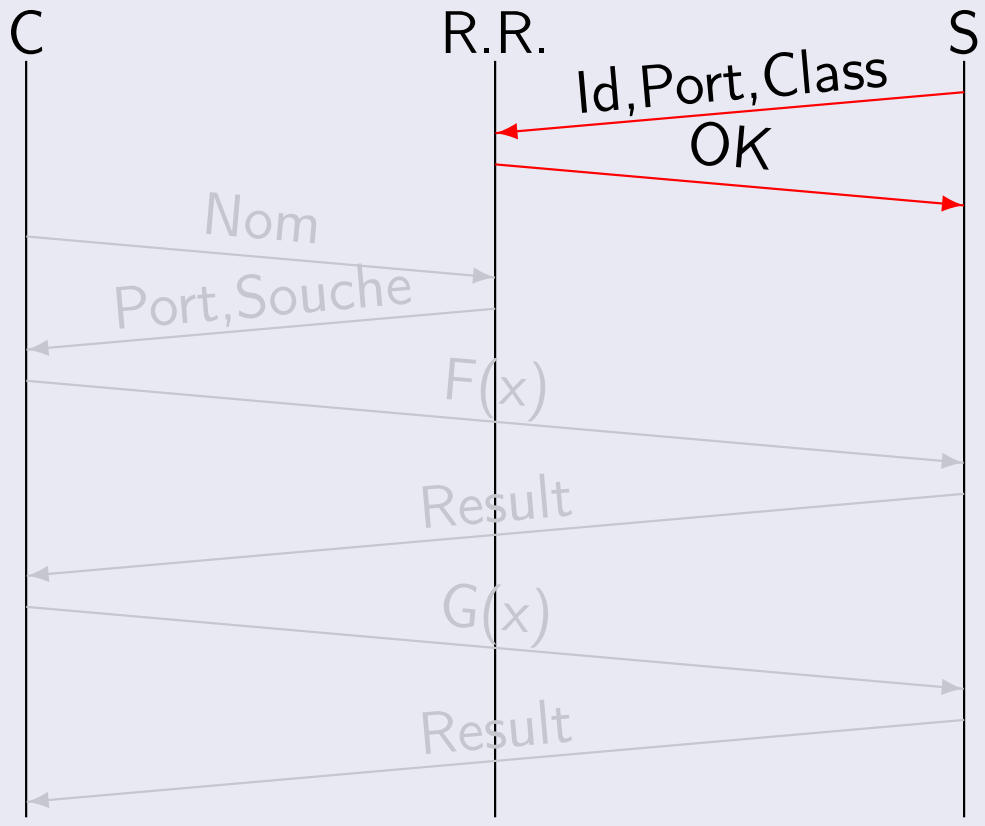
Génération d'un client

- Se connecter au serveur de localisation
  - ⇒ les parametre réseau du serveur
  - ⇒ la souche cliente
- Utiliser souche cliente

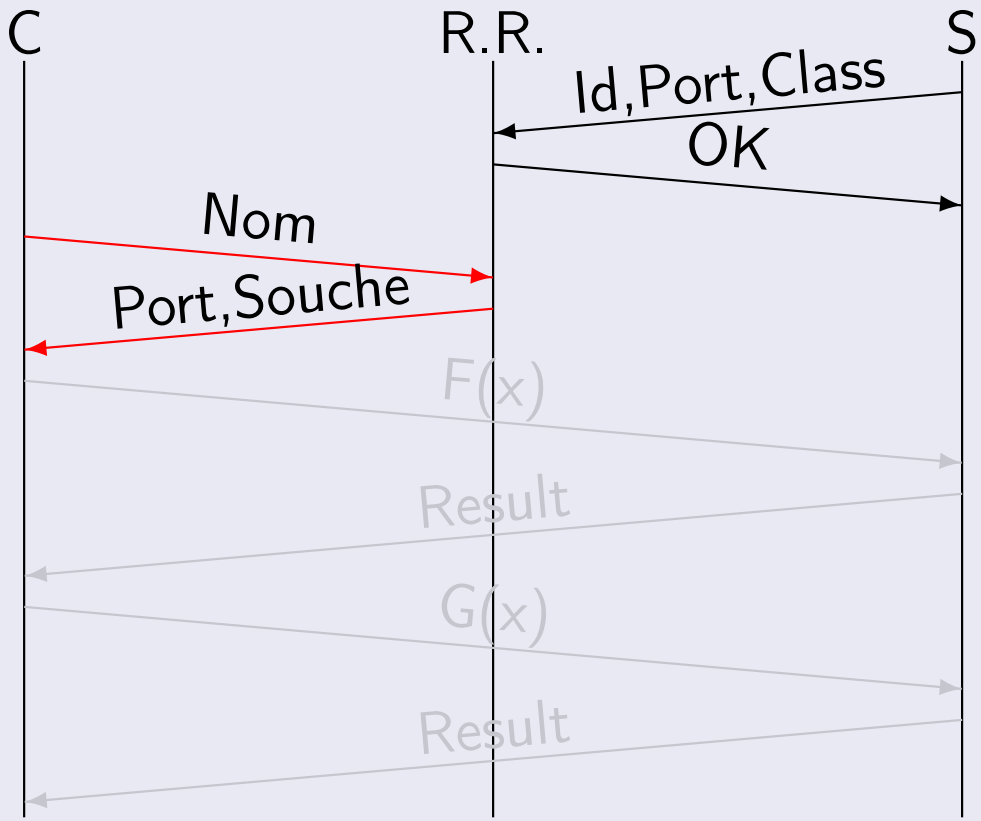
MSC



MSC



MSC



MSC

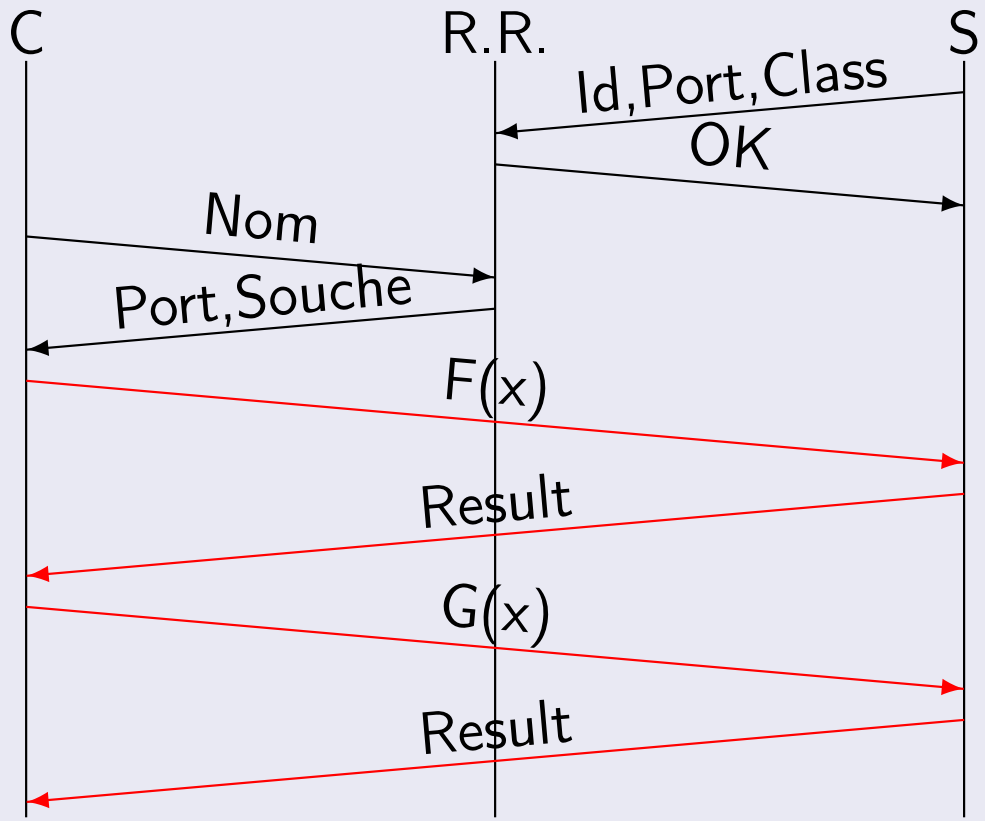


Table des matières

Interface

finance_ansver.java On a besoin du type réponse.

```
class finance_answer
    implements java.io.Serializable {
  public finance_answer() { values= new double[10]; }
  public int status;
  public double [] values;
}
```

finance.java L'interface avec les 2 fonctions du serveur.

```
import java.rmi.*;

public interface finance extends java.rmi.Remote {
  public finance_answer getcurr(String indice)
                throws RemoteException;
  public finance_answer getlast(String indice)
                throws RemoteException;
}
```

La classe serveur: finance_export.java

```
public class finance_export extends UnicastRemoteObject
                            implements finance {

  public finance_answer getcurr(String indice) {
    finance_answer a = new finance_answer();
    if (indice.equals("CAC40") ) {
        a.status = 0;
        a.values[0] = 3900;
    } else if ( indice.equals("NIKKEI") ) {
        ...
    } else {
        a.status = -1;
    }
    return a;
  }

  public finance_answer getlast(String indice) {
    ...
  }
}
```

La classe serveur: finance_export.java

```
public class finance_export extends UnicastRemoteObject
                            implements finance {

  public finance_answer getcurr(String indice) {
    finance_answer a = new finance_answer();
    if (indice.equals("CAC40") ) {
        a.status = 0;
        a.values[0] = 3900;
    } else if ( indice.equals("NIKKEI") ) {
        ...
    } else {
        a.status = -1;
    }
    return a;
  }

  public finance_answer getlast(String indice) {
    ...
  }
}
```

La classe serveur: finance_export.java

```
public class finance_export extends UnicastRemoteObject
                            implements finance {

  public finance_answer getcurr(String indice) {
    finance_answer a = new finance_answer();
    if (indice.equals("CAC40") ) {
        a.status = 0;
        a.values[0] = 3900;
    } else if ( indice.equals("NIKKEI") ) {
        ...
    } else {
        a.status = -1;
    }
    return a;
  }

  public finance_answer getlast(String indice) {
    ...
  }
}
```

La classe serveur: finance_export.java

```
public class finance_export extends UnicastRemoteObject
                            implements finance {

  public finance_answer getcurr(String indice) {
    finance_answer a = new finance_answer();
    if (indice.equals("CAC40") ) {
        a.status = 0;
        a.values[0] = 3900;
    } else if ( indice.equals("NIKKEI") ) {
        ...
    } else {
        a.status = -1;
    }
    return a;
  }

  public finance_answer getlast(String indice) {
    ...
  }
}
```

La classe serveur: finance_export.java

```
public class finance_export extends UnicastRemoteObject
                            implements finance {

  public finance_answer getcurr(String indice) {
    finance_answer a = new finance_answer();
    if (indice.equals("CAC40") ) {
        a.status = 0;
        a.values[0] = 3900;
    } else if ( indice.equals("NIKKEI") ) {
        ...
    } else {
        a.status = -1;
    }
    return a;
  }

  public finance_answer getlast(String indice) {
    ...
  }
}
```

La classe serveur: finance_export.java

```
public class finance_export extends UnicastRemoteObject
                            implements finance {

  public finance_answer getcurr(String indice) {
  }

  public finance_answer getlast(String indice) {
  }
}
```

Le serveur: finance_server.java

```
public class finance_server {
  public static void main(String[] args) throws Exception {
    finance_export o = new finance_export();
    Naming.rebind("myobj", o);
  }
}
```

La classe serveur: finance_export.java

```
public class finance_export extends UnicastRemoteObject
                            implements finance {

  public finance_answer getcurr(String indice) {
  }

  public finance_answer getlast(String indice) {
  }
}
```

Le serveur: finance_server.java

```
public class finance_server {
  public static void main(String[] args) throws Exception {
    finance_export o = new finance_export();
    Naming.rebind("myobj", o);
  }
}
```

La classe serveur: finance_export.java

```
public class finance_export extends UnicastRemoteObject
                            implements finance {

  public finance_answer getcurr(String indice) {
  }

  public finance_answer getlast(String indice) {
  }
}
```

Le serveur: finance_server.java

```
public class finance_server {
  public static void main(String[] args) throws Exception {
    finance_export o = new finance_export();
    Naming.rebind("myobj", o);
  }
}
```

Génération

```
⟶ javac finance_answer.java
⟶ javac finance.java
⟶ javac finance_export.java
⟶ javac finance_server.java
```

Lancement

```
⟶ rmiregistry &
⟶ java finance_server
```

rmiregistry doit pouvoir accéder à finance.class.

Code (fichier: finance_client.java.tex)

```
public class finance_client {
  public static void main(String[] args) throws Exception {

    // obtention d'un serveur
    String  url = "rmi://" + args[0] + "/" + args[1];
    finance f = (finance) Naming.lookup(url);

    // requête RPC
    finance_answer a;
    a = f.getcurr("CAC40");

    // utilisation de la réponse
    System.out.println("sta= " + a.status);
    System.out.println("val= " + a.values[0]);
   }
}
```

format d'une url:

```
rmi://<nom-serveur>:<port>/<nom-objet>
```

Code (fichier: finance_client.java.tex)

```
public class finance_client {
  public static void main(String[] args) throws Exception {

    // obtention d'un serveur
    String  url = "rmi://" + args[0] + "/" + args[1];
    finance f = (finance) Naming.lookup(url);

    // requête RPC
    finance_answer a;
    a = f.getcurr("CAC40");

    // utilisation de la réponse
    System.out.println("sta= " + a.status);
    System.out.println("val= " + a.values[0]);
   }
}
```

format d'une url:

rmi://<nom-serveur>:<port>/<nom-objet>

Code (fichier: finance_client.java.tex)

```
public class finance_client {
  public static void main(String[] args) throws Exception {

    // obtention d'un serveur
    String  url = "rmi://" + args[0] + "/" + args[1];
    finance f = (finance) Naming.lookup(url);

    // requête RPC
    finance_answer a;
    a = f.getcurr("CAC40");

    // utilisation de la réponse
    System.out.println("sta= " + a.status);
    System.out.println("val= " + a.values[0]);
   }
}
```

format d'une url:

rmi://<nom-serveur>:<port>/<nom-objet>

Code (fichier: finance_client.java.tex)

```
public class finance_client {
  public static void main(String[] args) throws Exception {

    // obtention d'un serveur
    String  url = "rmi://" + args[0] + "/" + args[1];
    finance f = (finance) Naming.lookup(url);

    // requête RPC
    finance_answer a;
    a = f.getcurr("CAC40");

    // utilisation de la réponse
    System.out.println("sta= " + a.status);
    System.out.println("val= " + a.values[0]);
   }
}
```

format d'une url:

rmi://<nom-serveur>:<port>/<nom-objet>

RMI: Remote Method Invocation: Implantation d'un client

Génération

```
⟶ javac finance_answer.java
⟶ javac finance.java
⟶ javac finance_client.java
```

Lancement

```
⟶ java finance_client nom-serveur myobj
```

RMI: Remote Method Invocation: Implantation d'un client

Génération

```
⟶ javac finance_answer.java
⟶ javac finance.java
⟶ javac finance_client.java
```

Lancement

```
⟶ java finance_client nom-serveur myobj
```

Table des matières

Vue programmation:

1) Définition des classes

2) Création d'objets

3) Initialisation

4) Algoritme

Vue programmation: OO classique

1) Définition des classes

```
class personne {
  string nom();
class parent : personne {
  personne* enfant(int i);
class enfant : personne {
  parent* pere();
  parent* mere();
```

2) Création d'objets

```
x= new ...
y= new ...
z= new ...
```

3) Initialisation

```
enfant*   penf;
penf= x;
```

4) Algoritme

```
personne* p;
personne* pere;
pere= penf->pere();
i=0;
while (pere->enfant(i)) {
  p= pere->enfant(i++);
  printf("%s\n",p->nom());
}
```

Vue programmation: OO middleware

1) Définition des classes

```
class personne {
  string nom();
class parent : personne {
  personne* enfant(int i);
class enfant : personne {
  parent* pere();
  parent* mere();
```

2) Création d'objets

```
P1       | P2     | P3
x= new   |        |
         |y= new  |
         |        | z=new
```

3) Initialisation

```
enfant*   penf;
penf= ...
```

4) Algoritme

```
personne* p;
personne* pere;
pere= penf->pere();
i=0;
while (pere->enfant(i)) {
  p= pere->enfant(i++);
  printf("%s\n",p->nom());
}
```

- Serveur crée des objets utilisables par les clients.
- Client utilise les objets fournis par les serveurs.
- Limitations

  Les objets sont des interfaces, pas d'attribut donnée d'où:

OO local	OO middleware
`obj->v1= 0;`	interdit
`obj->ptr= obj2;`	interdit

- Principe

  création de serveur d'interface

  création de classe cliente
- Middleware

  les paramètres réseaux sont complètement masqués

  multi-langages: clients et serveurs peuvent être écrits dans des langages différents.

Environnement

Définition d'un ensemble d'utilitaires pour aider le développement et la programmation d'applications réparties.

- Recherche d'un objet: page blanche, page jaune.

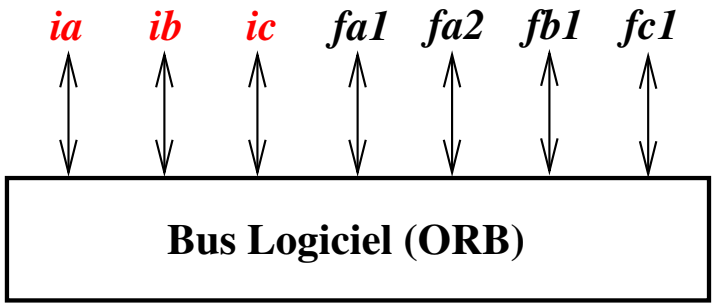
OO local	OO corba
`class X* obj;`	`class X* obj;`
`obj= new X();`	`obj= quelques_lignes_corba`

- Moteur transactionnel:
- Gestionnaire de persistence:
- Gestionnaire de propriétés:
- ...

$\Longrightarrow$ Ces utilitaires sont appelés les services.

$\Longrightarrow$ Ces services sont des objets CORBA.

ORB: Object Request Broker



2 types d'objets sur le bus

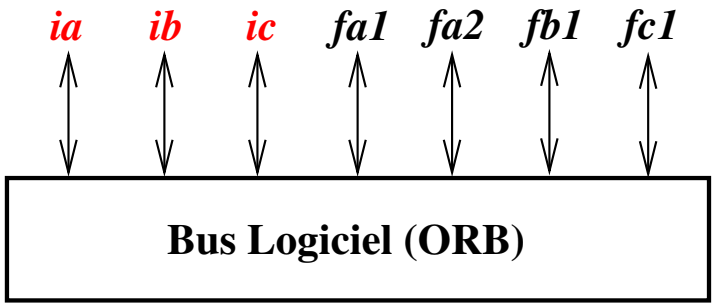
ia ib ic objets réels (instances)

- ils peuvent être dans le même processus ou dans plusieurs.
- ces processus sont dit "serveur"
- les données de l'objet sont stockées dans le processus

fa1 fa2 fb1 fc1 référence réseau à ia, ib, ic (*fantomes*)

- ils peuvent être dans le même processus ou dans plusieurs.
- ces processus sont dits "clients".
- ces objets *fantomes* n'ont pas de données, ce sont des interfaces fonctionnelles.

ORB: Object Request Broker



2 types d'objets sur le bus

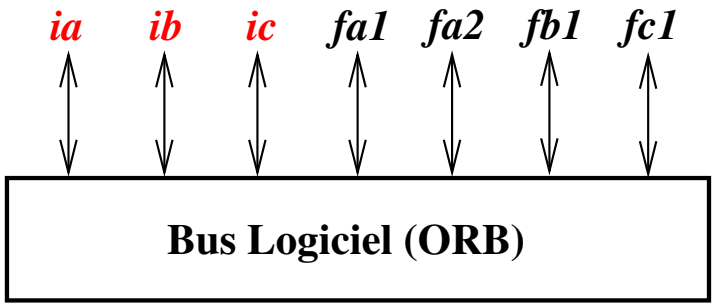
ia ib ic objets réels (instances)

- ils peuvent être dans le même processus ou dans plusieurs.
- ces processus sont dit "serveur"
- les données de l'objet sont stockées dans le processus

fa1 fa2 fb1 fc1 référence réseau à ia, ib, ic (*fantomes*)

- ils peuvent être dans le même processus ou dans plusieurs.
- ces processus sont dits "clients".
- ces objets *fantomes* n'ont pas de données, ce sont des interfaces fonctionnelles.

ORB: Object Request Broker



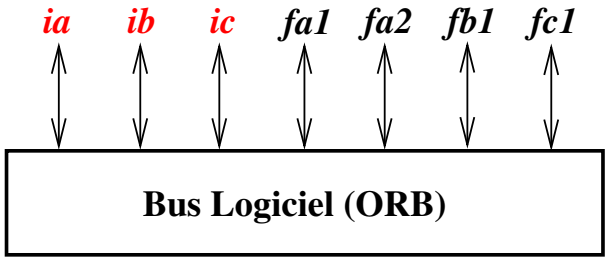
2 types d'objets sur le bus

ia ib ic objets réels (instances)

- ils peuvent être dans le même processus ou dans plusieurs.
- ces processus sont dit "serveur"
- les données de l'objet sont stockées dans le processus

fa1 fa2 fb1 fc1 référence réseau à ia, ib, ic (*fantomes*)

- ils peuvent être dans le même processus ou dans plusieurs.
- ces processus sont dits "clients".
- ces objets *fantomes* n'ont pas de données, ce sont des interfaces fonctionnelles.

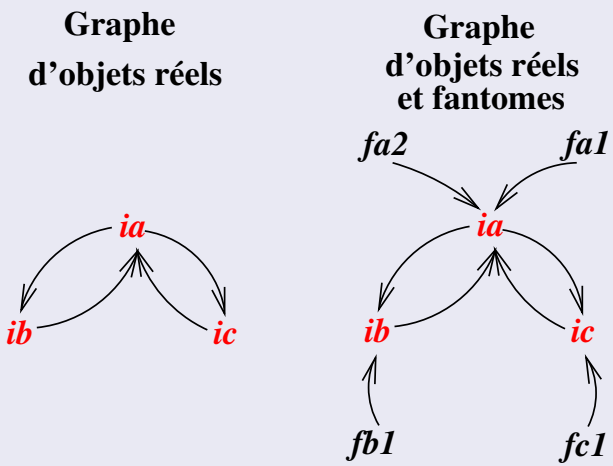


Graphe d'objets

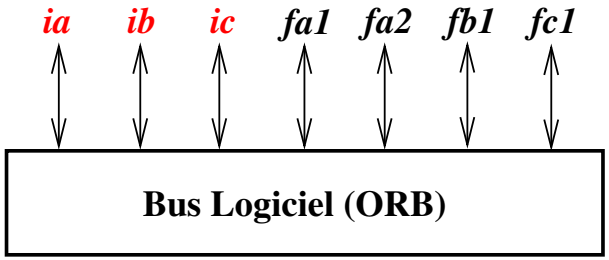
les objets fantômes sont des références sur les objets réels.

Répartition des objets

toutes les répartitions des objets sur les processus sont posibles.



ORB: Object Request Broker



Graphe d'objets

les objets fantômes sont des références sur les objets réels.

Répartition des objets

toutes les répartitions des objets sur les processus sont posibles.

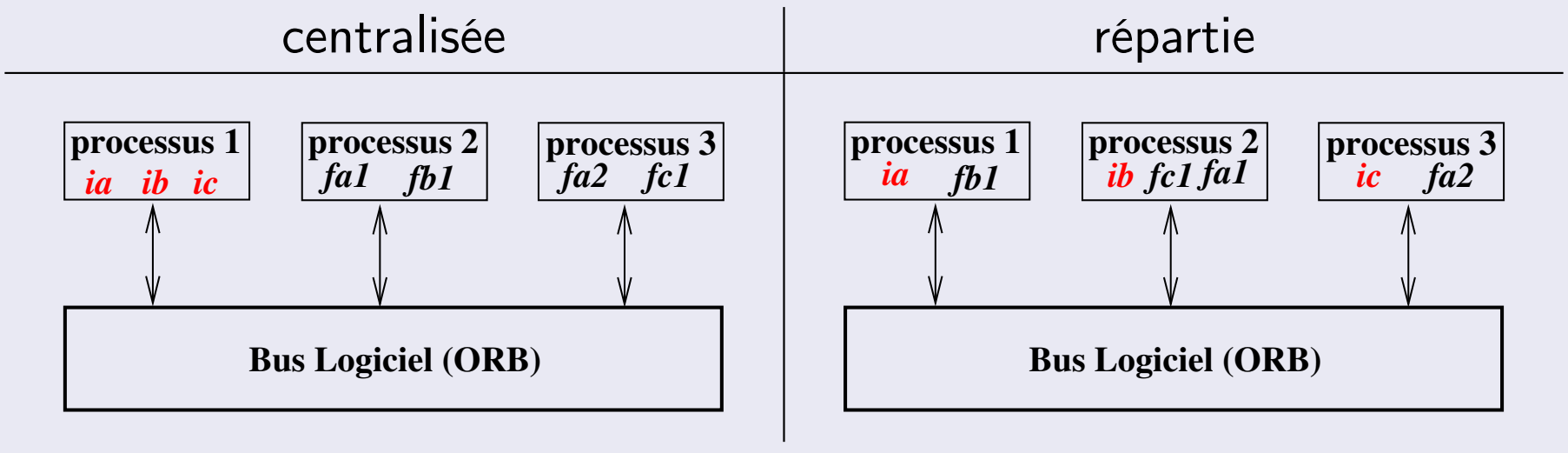


Table des matières

Eléments CORBA

Conceptuel

- IDL: Interface Description Langage
- IOR: Interface Object Reference

Logiciel

- compilateurs d'IDL vers différents langages
- bibliothèques d'objets contenant les souches clientes des utilitaires CORBA
- bibliothèques des serveurs des utilitaires CORBA

Génération d'une application CORBA

1. Ecrire la description IDL.
2. La compiler vers un langage.
3. Ecrire le client en utilisant les classes de la souche cliente, puis compiler le tout pour obtenir le client.
4. Compléter le squelette du serveur puis le compiler avec la squelette du serveur.

Eléments CORBA

Conceptuel

- IDL: Interface Description Langage
- IOR: Interface Object Reference

Logiciel

- compilateurs d'IDL vers différents langages
- bibliothèques d'objets contenant les souches clientes des utilitaires CORBA
- bibliothèques des serveurs des utilitaires CORBA

Génération d'une application CORBA

1. Ecrire la description IDL.
2. La compiler vers un langage.
3. Ecrire le client en utilisant les classes de la souche cliente, puis compiler le tout pour obtenir le client.
4. Compléter le squelette du serveur puis le compiler avec la squelette du serveur.

Eléments CORBA

Conceptuel

- IDL: Interface Description Langage
- IOR: Interface Object Reference

Logiciel

- compilateurs d'IDL vers différents langages
- bibliothèques d'objets contenant les souches clientes des utilitaires CORBA
- bibliothèques des serveurs des utilitaires CORBA

Génération d'une application CORBA

1. Ecrire la description IDL.
2. La compiler vers un langage.
3. Ecrire le client en utilisant les classes de la souche cliente, puis compiler le tout pour obtenir le client.
4. Compléter le squelette du serveur puis le compiler avec la squelette du serveur.

Etapes de conception

1. Description IDL
2. Génération de la souche cliente et du squelette du serveur
3. Implantation d'un serveur
4. Implantation d'un client
5. Mise en oeuvre

Fichier: name.idl

```
interface personne {
  string nom();
};
```

Principales caractéristiques de l'IDL

- héritage simple et multiple
- pas de redéfinition
- pas de surcharge
- pas de conflit de nom
- pas de membre données

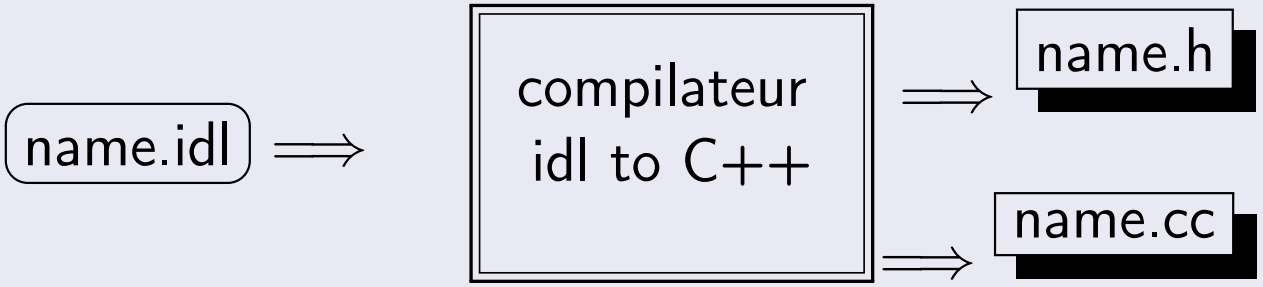
Fichier: name.idl

```
interface personne {
  string nom();
};
```

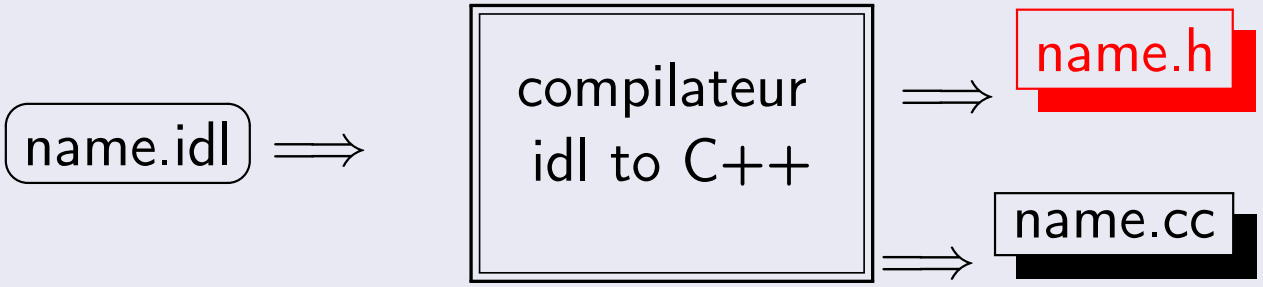
Principales caractéristiques de l'IDL

- héritage simple et multiple
- pas de redéfinition
- pas de surcharge
- pas de conflit de nom
- pas de membre données

Compilation de l'IDL (vers $C++$)



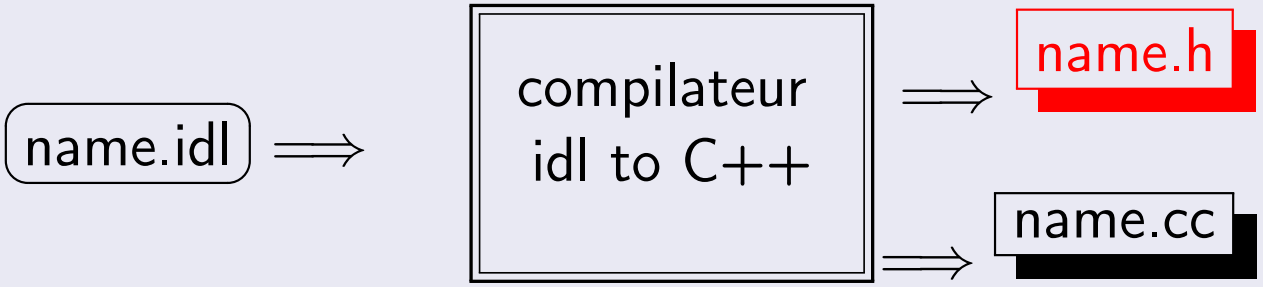
Compilation de l'IDL (vers $C++$)



Souche cliente

```
class personne : public CORBA::Object {
  char* nom();
}
```

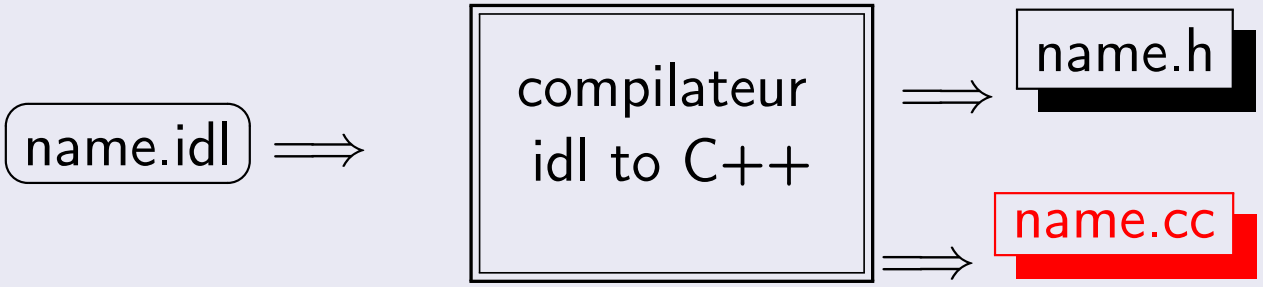
Compilation de l'IDL (vers $C++$)



Squelette du serveur

```
class POA_personne :
    public PortableServer::StaticImplementation {
  ...
  virtual char* nom() = 0;
  ...
}
```

Compilation de l'IDL (vers $C++$)



name.cc

- Implantation de la souche cliente (classe personne)
- Implantation du squelette du serveur (classe POA_personne)

Algorithme général

1. Définir une classe implantant les fonctions abstraites du squelette du serveur
2. Initialisation d'un ORB serveur
3. Créer des objets
4. Mettre les IOR des objets créés quelque part
5. Lancer le serveur

Squelette du serveur

```
class POA_personne :
    public PortableServer::StaticImplementation {
  ...
  virtual char* nom() = 0;
  ...
}
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};
```

Squelette du serveur

```
class POA_personne :
    public PortableServer::StaticImplementation {
  ...
  virtual char* nom() = 0;
  ...
}
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};

int main(int argc, char** argv) {
    CORBA::ORB_var orb = ...
    PortableServer::POA_var poa = ...


    ...
}
```

```
    CORBA::Object* obj;
    obj = orb->resolve_initial_references("RootPOA");
    poa= PortableServer::POA::_narrow(obj);
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};

int main(int argc, char** argv) {
    CORBA::ORB_var orb = CORBA::ORB_init(argc,argv);
    PortableServer::POA_var poa = ...


    ...
}
```

```
    CORBA::Object* obj;
    obj = orb->resolve_initial_references("RootPOA");
    poa= PortableServer::POA::_narrow(obj);
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};

int main(int argc, char** argv) {
    CORBA::ORB_var orb = CORBA::ORB_init(argc,argv);
    PortableServer::POA_var poa = ...


    ...
}
```

```
    CORBA::Object* obj;
    obj = orb->resolve_initial_references("RootPOA");
    poa= PortableServer::POA::_narrow(obj);
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};

int main(int argc, char** argv) {
    CORBA::ORB_var orb = CORBA::ORB_init(argc,argv);
    PortableServer::POA_var poa = ...


    ...
}
```

```
    CORBA::Object* obj;
    obj = orb->resolve_initial_references("RootPOA");
    poa= PortableServer::POA::_narrow(obj);
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};

int main(int argc, char** argv) {
    CORBA::ORB_var orb = CORBA::ORB_init(argc,argv);
    PortableServer::POA_var poa = ...

    ivan* i = new ivan();

    ...
}
```

```
    CORBA::Object* obj;
    obj = orb->resolve_initial_references("RootPOA");
    poa= PortableServer::POA::_narrow(obj);
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};

int main(int argc, char** argv) {
    CORBA::ORB_var orb = ...
    PortableServer::POA_var poa = ...

    ivan* i = new ivan();

    CORBA::Object* fi = i->_this();
    char* ior = orb->object_to_string(fi);
    printf("%s\n",ior);




}
```

Fichier: ivan.cc

```
class ivan : public POA_personne {
    ivan() {}
    char* nom() { return strdup("ivan"); }
};

int main(int argc, char** argv) {
    CORBA::ORB_var orb = ...
    PortableServer::POA_var poa = ...

    ivan* i = new ivan();

    CORBA::Object* fi = i->_this();
    char* ior = orb->object_to_string(fi);
    printf("%s\n",ior);

    fprintf(stderr,"Serveur pret\n");
    poa->the_POAManager()->activate();
    orb->run();
}
```

1. Initialisation d'un ORB client
2. Créer des objets fantômes
3. Utiliser les objets fantômes

souche cliente

```
class personne : public CORBA::Object {
  ...
  char* nom();
  ...
}
```

1. Initialisation d'un ORB client
2. Créer des objets fantômes
3. Utiliser les objets fantômes

souche cliente

```
class personne : public CORBA::Object {
  ...
  char* nom();
  ...
}
```

Fichier name.h: souche cliente

```
class personne : public CORBA::Object {
  char* nom();
}
```

Fichier client.cc

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  char* ior = argv[1];
  CORBA::Object* o = orb->string_to_object(ior);
  personne* f = personne::_narrow(o);
  if ( f == 0 ) { ... exit(1); }

  char* n = f->nom();
  printf("nom=%s\n",n);

  return 0;
}
```

Fichier name.h: souche cliente

```
class personne : public CORBA::Object {
  char* nom();
}
```

Fichier client.cc

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  char* ior = argv[1];
  CORBA::Object* o = orb->string_to_object(ior);
  personne* f = personne::_narrow(o);
  if ( f == 0 ) { ... exit(1); }

  char* n = f->nom();
  printf("nom=%s\n",n);

  return 0;
}
```

Fichier name.h: souche cliente

```
class personne : public CORBA::Object {
  char* nom();
}
```

Fichier client.cc

```
int main(int argc, char** argv) {

  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  char* ior = argv[1];
  CORBA::Object* o = orb->string_to_object(ior);
  personne* f = personne::_narrow(o);
  if ( f == 0 ) { ... exit(1); }

  char* n = f->nom();
  printf("nom=%s\n",n);

  return 0;
}
```

Fichier name.h: souche cliente

```
class personne : public CORBA::Object {
  char* nom();
}
```

Fichier client.cc

```
int main(int argc, char** argv) {

  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  char* ior = argv[1];
  CORBA::Object* o = orb->string_to_object(ior);
  personne* f = personne::_narrow(o);
  if ( f == 0 ) { ... exit(1); }

  char* n = f->nom();
  printf("nom=%s\n",n);

  return 0;
}
```

Fichier personne.java: souche cliente

```
public interface personne extends personneOperations ... {}
public interface personneOperations {
  String nom ();
}
```

Fichier client.java

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);

    String ior = args[0];
    org.omg.CORBA.Object o = orb.string_to_object(ior);
    personne f = personneHelper.narrow(o);

    String n = f.nom();
    System.out.println("nom=" + n);

 }
}
```

Fichier personne.java: souche cliente

```
public interface personne extends personneOperations ... {}
public interface personneOperations {
  String nom ();
}
```

Fichier client.java

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);

    String ior = args[0];
    org.omg.CORBA.Object o = orb.string_to_object(ior);
    personne f = personneHelper.narrow(o);

    String n = f.nom();
    System.out.println("nom=" + n);

 }
}
```

Fichier personne.java: souche cliente

```
public interface personne extends personneOperations ... {}
public interface personneOperations {
  String nom ();
}
```

Fichier client.java

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);

    String ior = args[0];
    org.omg.CORBA.Object o = orb.string_to_object(ior);
    personne f = personneHelper.narrow(o);

    String n = f.nom();
    System.out.println("nom=" + n);

 }
}
```

Fichier personne.java: souche cliente

```
public interface personne extends personneOperations ... {}
public interface personneOperations {
  String nom ();
}
```

Fichier client.java

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);

    String ior = args[0];
    org.omg.CORBA.Object o = orb.string_to_object(ior);
    personne f = personneHelper.narrow(o);

    String n = f.nom();
    System.out.println("nom=" + n);

 }
}
```

serveur: (name.idl, ivan.cc)

```
⟶ idl name.idl && ls
⟶ name.cc name.h name.idl ivan.cc
⟶ g++ -I ... ivan.cc name.cc -L ... -l... -l... -o
serv
⟶ ./serv
IOR:0100000011000000494....
Serveur pret
```

client: (name.idl client.cc)

```
⟶ idl name.idl && ls
name.idl client.cc name.cc name.h
⟶ g++ -I ... client.cc name.cc -L ... -l... -l... -o
cli
⟶ ./cli IOR:0100000011000000494....
nom=ivan
⟶
```

serveur: (name.idl, ivan.cc)

```
 ⟶ idl name.idl && ls
⟶ name.cc name.h name.idl ivan.cc
⟶ g++ -I ...  ivan.cc name.cc -L ...  -l...  -l...  -o
serv
⟶ ./serv
IOR:0100000011000000494....
Serveur pret
```

client: (name.idl client.cc)

```
 ⟶ idl name.idl && ls
name.idl client.cc name.cc name.h
⟶ g++ -I ...  client.cc name.cc -L ...  -l...  -l...  -o
cli
⟶ ./cli IOR:0100000011000000494....
nom=ivan
⟶
```

serveur: (name.idl, ivan.cc)

```
 ⟶ idl name.idl && ls
⟶ name.cc name.h name.idl ivan.cc
⟶ g++ -I ...  ivan.cc name.cc -L ...  -l...  -l...  -o
serv
⟶ ./serv
IOR:0100000011000000494....
Serveur pret
```

client: (name.idl client.cc)

```
 ⟶ idl name.idl && ls
name.idl client.cc name.cc name.h
⟶ g++ -I ...  client.cc name.cc -L ...  -l...  -l...  -o
cli
⟶ ./cli IOR:0100000011000000494....
nom=ivan
⟶
```

serveur: (name.idl, ivan.cc)

```
 ⟶ idl name.idl && ls
⟶ name.cc name.h name.idl ivan.cc
⟶ g++ -I ...  ivan.cc name.cc -L ...  -l...  -l...  -o
serv
⟶ ./serv
IOR:0100000011000000494....
Serveur pret
```

client: (name.idl client.cc)

```
 ⟶ idl name.idl && ls
name.idl client.cc name.cc name.h
⟶ g++ -I ...  client.cc name.cc -L ...  -l...  -l...  -o
cli
⟶ ./cli IOR:0100000011000000494....
nom=ivan
⟶
```

serveur: (name.idl, ivan.cc)

```
 ⟶ idl name.idl && ls
⟶ name.cc name.h name.idl ivan.cc
⟶ g++ -I ...  ivan.cc name.cc -L ...  -l...  -l...  -o
serv
⟶ ./serv
IOR:0100000011000000494....
Serveur pret
```

client: (name.idl client.cc)

```
 ⟶ idl name.idl && ls
name.idl client.cc name.cc name.h
⟶ g++ -I ...  client.cc name.cc -L ...  -l...  -l...  -o
cli
⟶ ./cli IOR:0100000011000000494....
nom=ivan
⟶
```

serveur: (name.idl, ivan.cc)

```
 ⟶ idl name.idl && ls
⟶ name.cc name.h name.idl ivan.cc
⟶ g++ -I ...  ivan.cc name.cc -L ...  -l...  -l...  -o
serv
⟶ ./serv
IOR:0100000011000000494....
Serveur pret
```

client: (name.idl client.cc)

```
 ⟶ idl name.idl && ls
name.idl client.cc name.cc name.h
⟶ g++ -I ...  client.cc name.cc -L ...  -l...  -l...  -o
cli
⟶ ./cli IOR:0100000011000000494....
nom=ivan
⟶
```

serveur: (name.idl, ivan.cc)

```
 ⟶ idl name.idl && ls
⟶ name.cc name.h name.idl ivan.cc
⟶ g++ -I ...  ivan.cc name.cc -L ...  -l...  -l...  -o
serv
⟶ ./serv
IOR:0100000011000000494....
Serveur pret
```

client: (name.idl client.java)

```
 ⟶ idlj name.idl && ls
name.idl client.java _personneStub.java personne.java ...
⟶ javac client.java
⟶ java client IOR:0100000011000000494....
nom=van
⟶
```

Fonctions principales

ior $\Longleftrightarrow$ objet-fantôme

```
CORBA::Object* f = orb->string_to_object(ior);
const char* ior = object_to_string(f);
```

objet-fantôme général $\Longrightarrow$ objet-fantôme spécialisé

```
CORBA::Object* f;
XXX_client* fs = XXX_client::_narrow(f);
```

objet-serveur $\Longrightarrow$ objet-fantôme

```
XXX_seveur* os;
XXX_client* fs = os->_this();
CORBA::Object* f = os->_this();
```

et par héritage

```
XXX_client* fs;
CORBA::Object* f = fs;
```

Table des matières

Ex. finance (centralisée): IDL

Code: (finnance.idl)

```
module MF {

  struct Tanswer {
      long                status;
      sequence<double> values;
  };

  interface finance {
      Tanswer getcurr(in string indice);
      Tanswer getlast(in string indice);
  };
};
```

Un peu plus d'IDL

- struct: Définition de types.
- sequence: CORBA définit des containeurs génériques.
  textcolorbluesequence est un tableau dynamique.
- module: encapsulation.

Ex. finance (centralisée): IDL

Code: (finnance.idl)

```
module MF {

  struct Tanswer {
      long                status;
      sequence<double> values;
  };

  interface finance {
      Tanswer getcurr(in string indice);
      Tanswer getlast(in string indice);
  };
};
```

Un peu plus d'IDL

- struct: Définition de types.
- sequence: CORBA définit des containeurs génériques. textcolorbluesequence est un tableau dynamique.
- module: encapsulation.

Ex. finance (centralisée): IDL

Code: (finnance.idl)

```
module MF {

  struct Tanswer {
      long                status;
      sequence<double> values;
  };

  interface finance {
      Tanswer getcurr(in string indice);
      Tanswer getlast(in string indice);
  };
};
```

Un peu plus d'IDL

- struct: Définition de types.
- sequence: CORBA définit des containeurs génériques. textcolorbluesequence est un tableau dynamique.
- module: encapsulation.

Ex. finance (centralisée): classe squelette

```
namespace POA_MF {

class finance :
  virtual public PortableServer::StaticImplementation {
  ...
  virtual ::MF::Tanswer* getcurr( const char* indice ) = 0;
  virtual ::MF::Tanswer* getlast( const char* indice ) = 0;
};
} // namespace
```

Ex. finance (centralisée): Une implantation de la classe squelette

```
class financeImp : public POA_MF::finance {
public:
  financeImp() {}
  ::MF::Tanswer* getcurr( const char* indice ) {
    ::MF::Tanswer* a = new Tanswer;
    if ( strcmp(*indice,"CAC40")==0 ) {
      a->status=0; a->values.length(1); a->values[0] = 3500;
    } else if ( strcmp(*indice,"NIKKEI")==0 ) {
      ...
    } else {
      a.status=-1;
    }
    return a;
  }
  ::MF::Tanswer* getlast( const char* indice ) {
  }
};
```

Ex. finance (centralisée): Qualités de cette version

Cette version est correcte et fonctionne mais:

- Le serveur central $\longrightarrow$ pas de répartition de la charge.
- Ajout d'indices $\longrightarrow$ arret du serveur.
- Evolution du logiciel $\longrightarrow$ plus en plus complexe $\longrightarrow$ de moins en moins modifiable
- Le serveur central $\Longrightarrow$ géré par 1 société $\Longrightarrow$ difficulté de regrouper toutes les sociétés du domaine.

bonne programation descendante quasi équivalente à la version RPC.

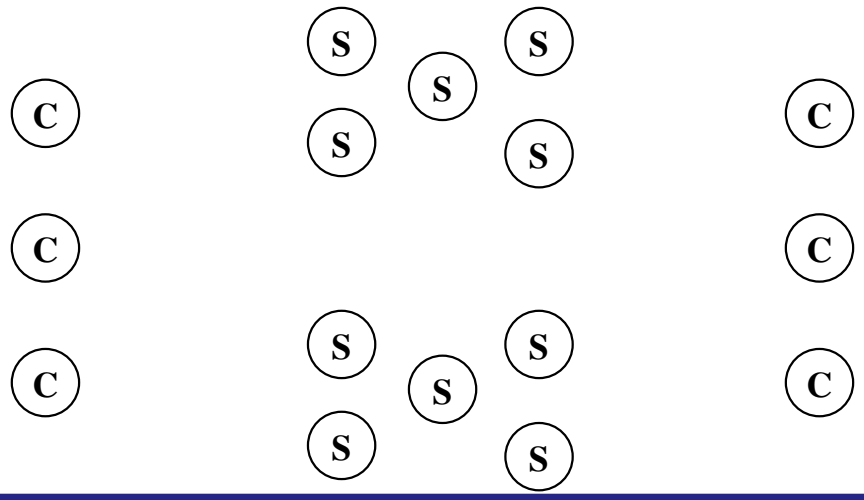
Ex. finance (centralisée): Qualités de cette version

Cette version est correcte et fonctionne mais:

- Le serveur central $\longrightarrow$ pas de répartition de la charge.
- Ajout d'indices $\longrightarrow$ arret du serveur.
- Evolution du logiciel $\longrightarrow$ plus en plus complexe $\longrightarrow$ de moins en moins modifiable
- Le serveur central $\Longrightarrow$ géré par 1 société $\Longrightarrow$ difficulté de regrouper toutes les sociétés du domaine.

bonne programation descendante quasi équivalente à la version RPC.

Une belle application répartie

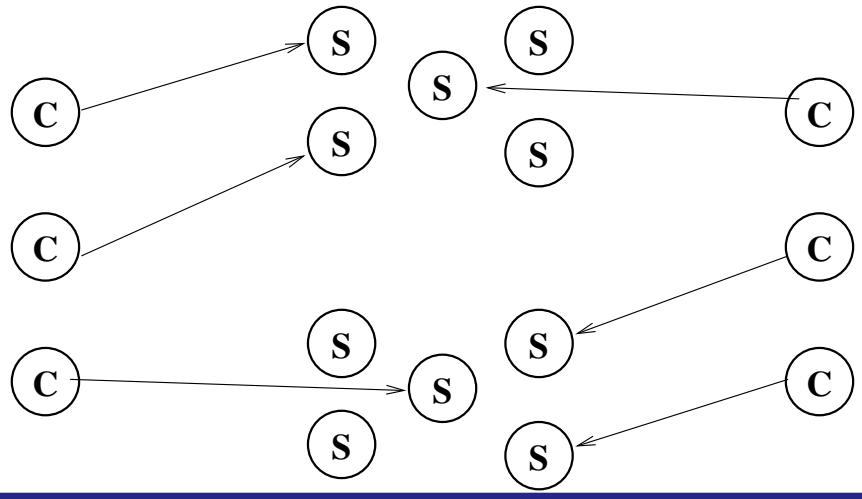


Elements

- des millions de serveurs
- des 100 millions de clients
- une glue: HTTP et HTML
- quelques GPS: google, ...

conception ascendante par agrégation d'éléments.

Une belle application répartie

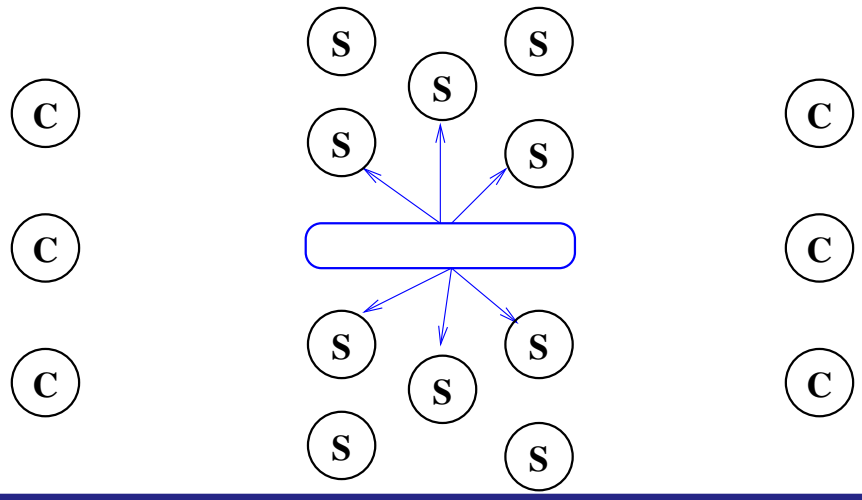


Elements

- des millions de serveurs
- des 100 millions de clients
- une glue: HTTP et HTML
- quelques GPS: google, ...

conception ascendante par agrégation d'éléments.

Une belle application répartie

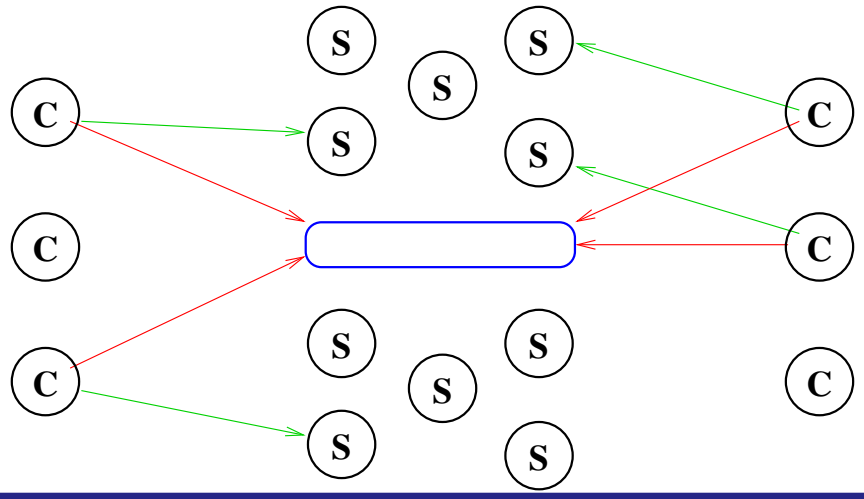


Elements

- des millions de serveurs
- des 100 millions de clients
- une glue: HTTP et HTML
- quelques GPS: google, ...

conception ascendante par agrégation d'éléments.

Une belle application répartie

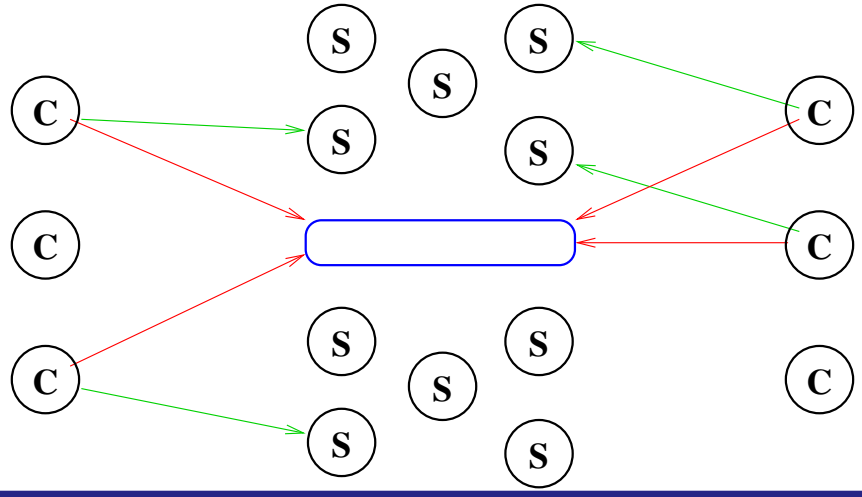


Elements

- des millions de serveurs
- des 100 millions de clients
- une glue: HTTP et HTML
- quelques GPS: google, ...

conception ascendante par agrégation d'éléments.

Une belle application répartie

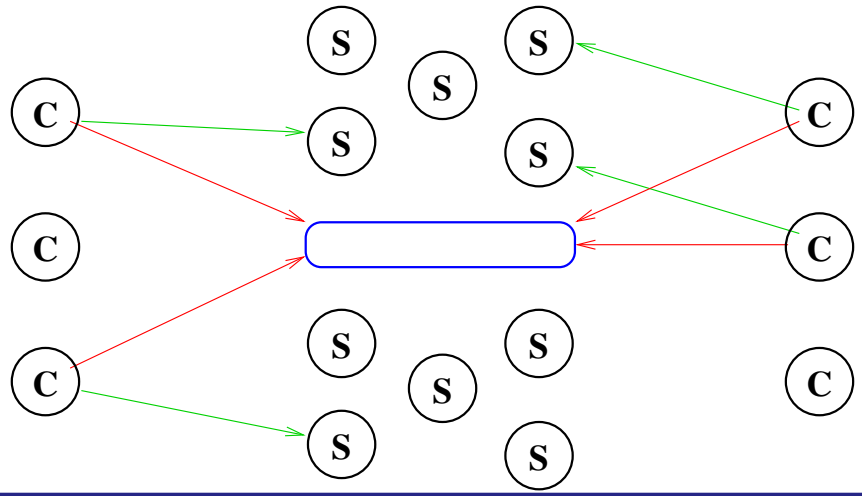


Avantages

- conception décentralisée (pas de maître d'oeuvre ou d'ouvrage).
- pas de contraines de développement (HTML, PHP, PYTHON, CGI, APPLET, ...).
- ajout de briques aisées.

conception ascendante par agrégation d'éléments.

Une belle application répartie



Avantages

- conception décentralisée (pas de maître d'oeuvre ou d'ouvrage).
- pas de contraines de développement (HTML, PHP, PYTHON, CGI, APPLET, ...).
- ajout de briques aisées.

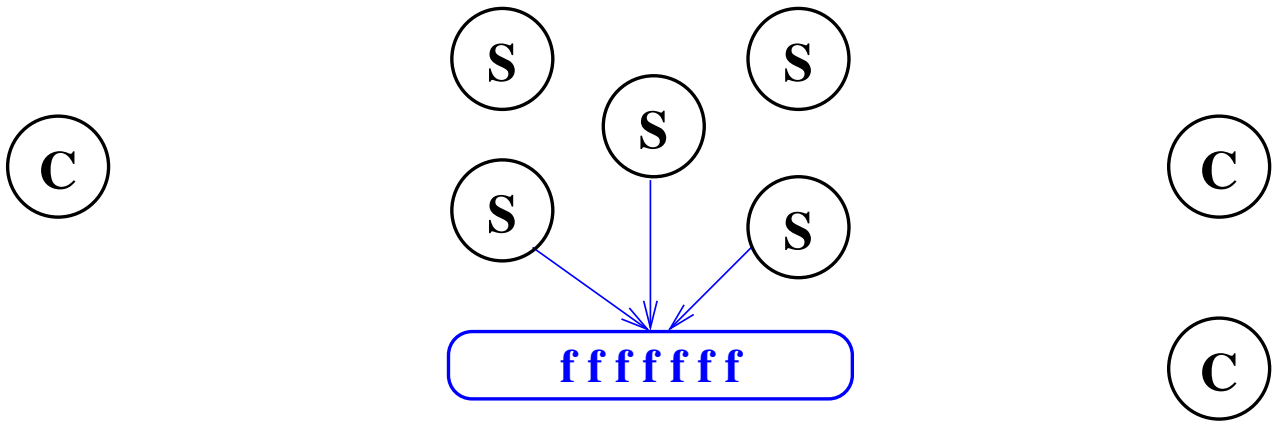
conception ascendante par agrégation d'éléments.

Principe

1. Un GPS
2. Les seveurs s'enregistrent dans le GPS .
3. Les clients demandent les serveus au GPS .
4. Les clients peuvent communiquer avec les serveurs .

Séparation complète de la localisation et du métier.

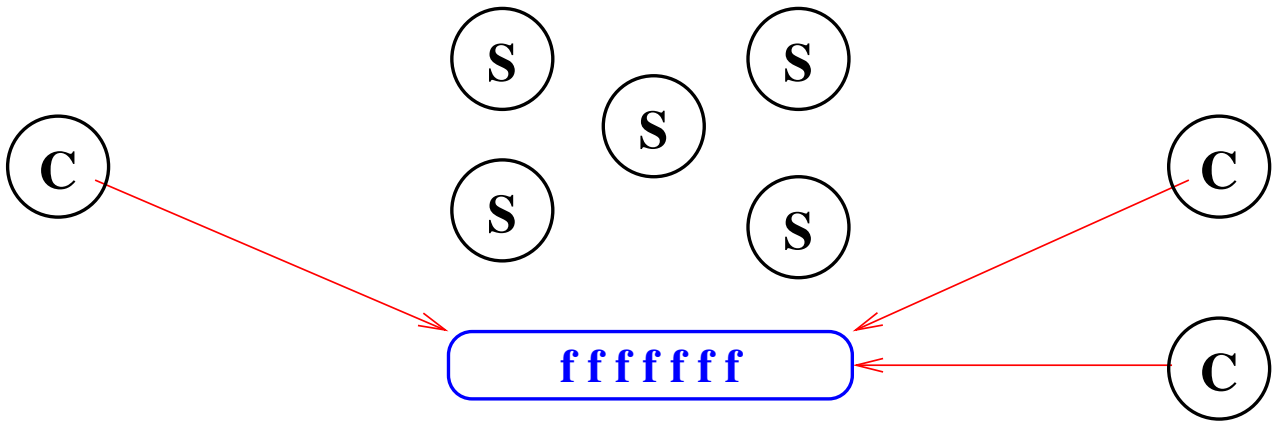
Principe



1. Un GPS
2. Les seveurs s'enregistrent dans le GPS .
3. Les clients demandent les serveus au GPS .
4. Les clients peuvent communiquer avec les serveurs .

Séparation complète de la localisation et du métier.

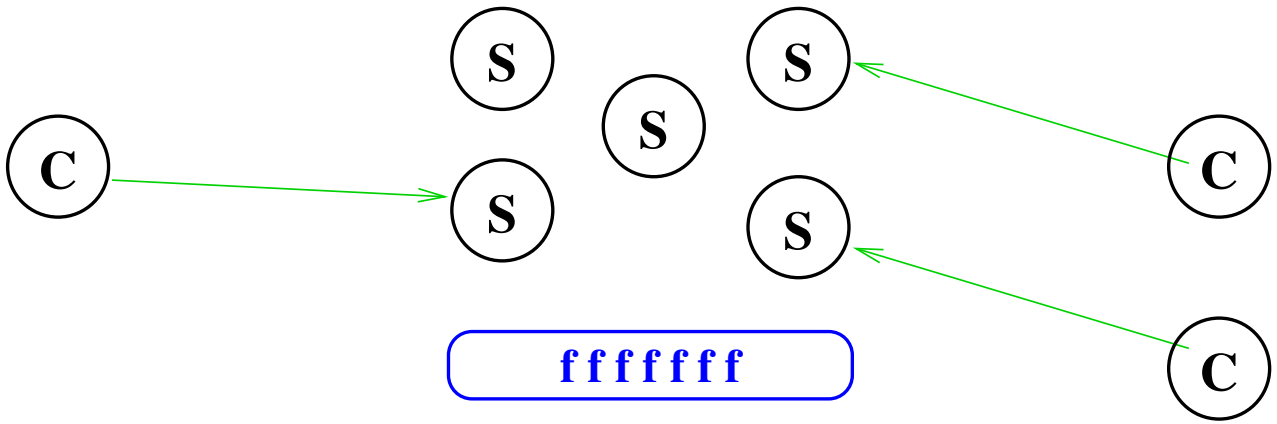
Principe



1. Un GPS
2. Les seveurs s'enregistrent dans le GPS .
3. Les clients demandent les serveus au GPS .
4. Les clients peuvent communiquer avec les serveurs .

Séparation complète de la localisation et du métier.

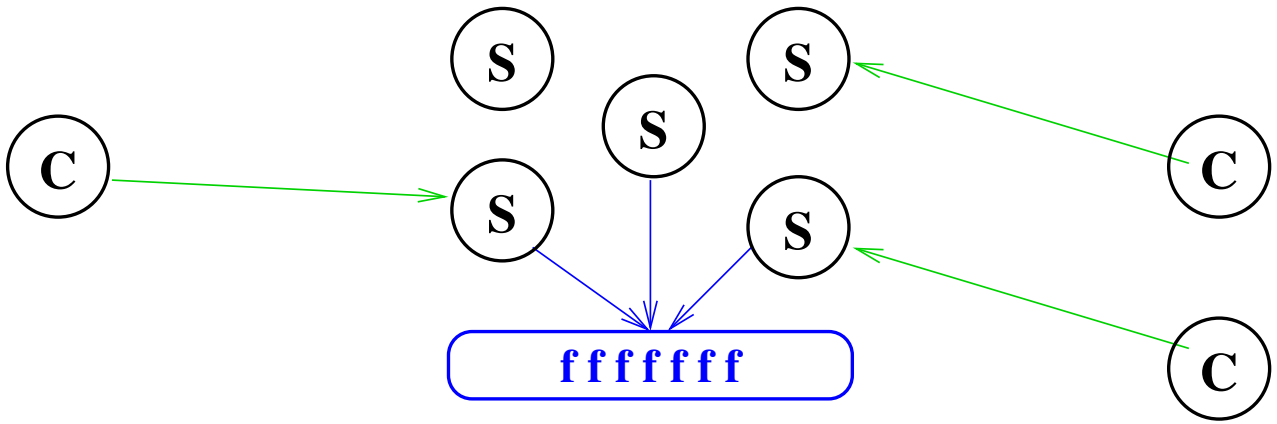
Principe



1. Un GPS
2. Les seveurs s'enregistrent dans le GPS .
3. Les clients demandent les serveus au GPS .
4. Les clients peuvent communiquer avec les serveurs .

Séparation complète de la localisation et du métier.

Principe



1. Un GPS
2. Les seveurs s'enregistrent dans le GPS .
3. Les clients demandent les serveus au GPS .
4. Les clients peuvent communiquer avec les serveurs .

Séparation complète de la localisation et du métier.

Ex. finance (répartie): IDL

Code: (finnance.idl)

```
typedef sequence<double> DoubleArray;

interface finance {
  long getcurr(out double value);
  long getlast(out DoubleArray values);
};

interface gps {
  finance get(in string indice);
  void    add(in string indice, in finance objet);
};
```

class finance plus besoin d'indice .

class gps une méthode d'enregistrement et un méthode de résolution

Ex. finance (répartie): IDL

Code: (finnance.idl)

```
typedef sequence<double> DoubleArray;

interface finance {
  long getcurr(out double value);
  long getlast(out DoubleArray values);
};

interface gps {
  finance get(in string indice);
  void    add(in string indice, in finance objet);
};
```

class finance plus besoin d'indice .

class gps une méthode d'enregistrement et un méthode de résolution

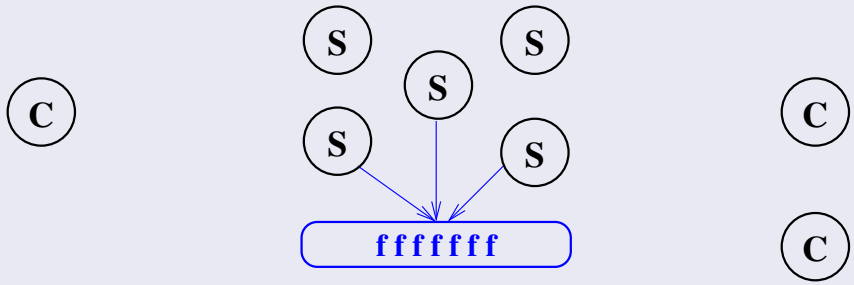
Ex. finance (répartie): IDL

Code: (finnance.idl)

```
typedef sequence<double> DoubleArray;

interface finance {
  long getcurr(out double value);
  long getlast(out DoubleArray values);
};

interface gps {
  finance get(in string indice);
  void    add(in string indice, in finance objet);
};
```

class finance plus besoin d'indice .

class gps une méthode d'enregistrement et un méthode de résolution



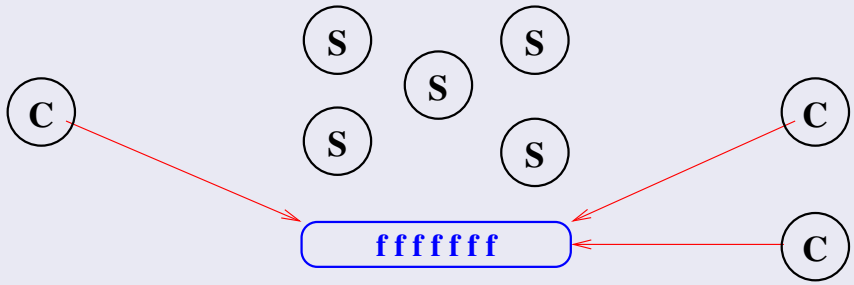
Ex. finance (répartie): IDL

Code: (finnance.idl)

```
typedef sequence<double> DoubleArray;

interface finance {
  long getcurr(out double value);
  long getlast(out DoubleArray values);
};

interface gps {
  finance get(in string indice);
  void    add(in string indice, in finance objet);
};
```

class finance plus besoin d'indice .

class gps une méthode d'enregistrement et un méthode de résolution



Ex. finance (répartie)- Serveur CAC40: Classe squelette (fichier: finance.h généré)

```
class POA_finance :
    virtual public PortableServer::StaticImplementation {
  ...
  virtual CORBA::Long getcurr( CORBA::Double_out value ) = 0;
  virtual CORBA::Long getlast( ::DoubleArray_out values ) = 0;
};
```

Ex. finance (répartie)- Serveur CAC40: Classe serveur (fichier: cac40.cc)

```
class Cac40 : public POA_finance {
  public:
  CORBA::Long getcurr(CORBA::Double& value) {
    value = 2000;
    return 0;
  }
  CORBA::Long getlast(::DoubleArray_out values) {
    values = new DoubleArray;
    values->length(3);
    (*values)[0]=2001; (*values)[1]=2002; (*values)[2]=2003;
    return 0;
  }
};
```

Est reduite au métier.

Ex. finance (répartie)- Serveur CAC40: Classe serveur (fichier: cac40.cc)

```
class Cac40 : public POA_finance {
  public:
  CORBA::Long getcurr(CORBA::Double& value) {
    value = 2000;
    return 0;
  }
  CORBA::Long getlast(::DoubleArray_out values) {
    values = new DoubleArray;
    values->length(3);
    (*values)[0]=2001; (*values)[1]=2002; (*values)[2]=2003;
    return 0;
  }
};
```

Est reduite au métier.

Ex. finance (répartie)- Serveur CAC40: Classe serveur (fichier: cac40.cc)

```
class Cac40 : public POA_finance {
  public:
  CORBA::Long getcurr(CORBA::Double& value) {
    value = 2000;
    return 0;
  }
  CORBA::Long getlast(::DoubleArray_out values) {
    values = new DoubleArray;
    values->length(3);
    (*values)[0]=2001; (*values)[1]=2002; (*values)[2]=2003;
    return 0;
  }
};
```

Est reduite au métier.

Ex. finance (répartie)- Serveur CAC40: Souche cliente du GPS (fichier: finance.h généré)

```
class gps : virtual public CORBA::Object {
   ...
   ::finance_ptr get( const char* indice );
   void          add( const char* indice, ::finance_ptr objet )
   ...
};
```

Ex. finance (répartie)- Serveur CAC40: Programme principal (fichier: cac40.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;
  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...
  // récupération d'un fantome du GPS
  char* ior=argv[1];
  f= ..........................
  gps* fpgs = ..................
  // mise sur l'orb d'un objet.
  Cac40* icac40 = new Cac40();
  // publication de l'ior de l'objet
  ....................................
  // lancement du serveur
  ...
}
```

Ex. finance (répartie)- Serveur CAC40: Programme principal (fichier: cac40.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;
  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...
  // récupération d'un fantome du GPS
  char* ior=argv[1];
  f= ..........................
  gps* fpgs = ..................
  // mise sur l'orb d'un objet.
  Cac40* icac40 = new Cac40();
  // publication de l'ior de l'objet
  ....................................
  // lancement du serveur
  ...
}
```

Ex. finance (répartie)- Serveur CAC40: Programme principal (fichier: cac40.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;

  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...

  // récupération d'un fantome du GPS
  char* ior=argv[1];
  f= ..........................
  gps* fpgs = ..................

  // mise sur l'orb d'un objet.
  Cac40* icac40 = new Cac40();

  // publication de l'ior de l'objet
  ....................................

  // lancement du serveur
  ...
}
```

Ex. finance (répartie)- Serveur CAC40: Programme principal (fichier: cac40.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;
  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...
  // récupération d'un fantome du GPS
  char* ior=argv[1];
  f= ..........................
  gps* fpgs = ..................
  // mise sur l'orb d'un objet.
  Cac40* icac40 = new Cac40();
  // publication de l'ior de l'objet
  ....................................
  // lancement du serveur
  ...
}
```

Ex. finance (répartie)- Serveur CAC40: Programme principal (fichier: cac40.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;
  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...
  // récupération d'un fantome du GPS
  char* ior=argv[1];
  f= ..........................
  gps* fpgs = ..................
  // mise sur l'orb d'un objet.
  Cac40* icac40 = new Cac40();
  // publication de l'ior de l'objet
  ....................................
  // lancement du serveur
  ...
}
```

Ex. finance (répartie)- Serveur GPS: Classe squelette (fichier: finance.h généré)

```
class POA_gps :
    virtual public PortableServer::StaticImplementation {
  ...
  virtual ::finance_ptr get( const char* indice ) = 0;
  virtual void add( const char* indice, ::finance_ptr objet )
  ...
};
```

Ex. finance (répartie)- Serveur GPS: Classe serveur (fichier: gps.cc)

Spécification - Structure de données - Constructeurs - Services

```
class gpsImpl : public POA_gps {
  int lookup(const char* indice) { ... }
  int             nb;
  char*           name[100];
  finance_ptr     objets[100];
  gpsImpl() : nb(0) {}
  virtual finance_ptr get(const char* indice)
  {
    int i = lookup(indice);
    return i<0 ? (finance_ptr)0
               : finance::_duplicate(objets[i]);
  }
  virtual void add(const char* indice, finance_ptr obj)
};
```

Ex. finance (répartie)- Serveur GPS: Classe serveur (fichier: gps.cc)

Spécification - Structure de données - Constructeurs - Services

```
class gpsImpl : public POA_gps {
  int lookup(const char* indice) { ... }
  int             nb;
  char*           name[100];
  finance_ptr     objets[100];
  gpsImpl() : nb(0) {}
  virtual finance_ptr get(const char* indice)
  {
    int i = lookup(indice);
    return i<0 ? (finance_ptr)0
               : finance::_duplicate(objets[i]);
  }
  virtual void add(const char* indice, finance_ptr obj)
};
```

Ex. finance (répartie)- Serveur GPS: Classe serveur (fichier: gps.cc)

Spécification - Structure de données - Constructeurs - Services

```
class gpsImpl : public POA_gps {
  int lookup(const char* indice) { ... }
  int             nb;
  char*           name[100];
  finance_ptr     objets[100];
  gpsImpl() : nb(0) {}
  virtual finance_ptr get(const char* indice)
  {
    int i = lookup(indice);
    return i<0 ? (finance_ptr)0
               : finance::_duplicate(objets[i]);
  }
  virtual void add(const char* indice, finance_ptr obj)
};
```

Ex. finance (répartie)- Serveur GPS: Classe serveur (fichier: gps.cc)

Spécification - Structure de données - Constructeurs - Services

```
class gpsImpl : public POA_gps {
  int lookup(const char* indice) { ... }
  int             nb;
  char*           name[100];
  finance_ptr     objets[100];
  gpsImpl() : nb(0) {}
  virtual finance_ptr get(const char* indice)
  {
    int i = lookup(indice);
    return i<0 ? (finance_ptr)0
               : finance::_duplicate(objets[i]);
  }
  virtual void add(const char* indice, finance_ptr obj)
};
```

Ex. finance (répartie)- Serveur GPS: Classe serveur (fichier: gps.cc)

Spécification - Structure de données - Constructeurs - Services

```
class gpsImpl : public POA_gps {
  int lookup(const char* indice) { ... }
  int             nb;
  char*           name[100];
  finance_ptr     objets[100];
  gpsImpl() : nb(0) {}
  virtual finance_ptr get(const char* indice)
  {
    int i = lookup(indice);
    return i<0 ? (finance_ptr)0
               : finance::_duplicate(objets[i]);
  }
  virtual void add(const char* indice, finance_ptr obj)
};
```

Ex. finance (répartie)- Serveur GPS: Classe serveur (fichier: gps.cc)

Spécification - Structure de données - Constructeurs - Services

```
class gpsImpl : public POA_gps {
  int lookup(const char* indice) { ...  }
  int             nb;
  char*           name[100];
  finance_ptr     objets[100];
  gpsImpl() : nb(0) {}
  virtual finance_ptr get(const char* indice)
  {
    int i = lookup(indice);
    return i<0 ? (finance_ptr)0
               : finance::_duplicate(objets[i]);
  }
  virtual void add(const char* indice, finance_ptr obj)
};
```

Ex. finance (répartie)- Serveur GPS: Classe serveur (fichier: gps.cc)

Spécification - Structure de données - Constructeurs - Services

```
class gpsImpl : public POA_gps {
  int lookup(const char* indice) { ...  }
  virtual finance_ptr get(const char* indice)
  {
    int i = lookup(indice);
    return i<0 ? (finance_ptr)0
               : finance::_duplicate(objets[i]);
  }
  virtual void add(const char* indice, finance_ptr obj)
  {
    int i = lookup(indice);
    if (i<0) {
      i=nb; nb+=1;
      name[i] = strdup(indice);
    }
    objets[i] = obj->_duplicate(obj);
  }
};
```

Ex. finance (répartie)- Serveur GPS: Classe serveur (fichier: gps.cc)

Spécification - Structure de données - Constructeurs - Services

```
class gpsImpl : public POA_gps {
  int lookup(const char* indice) { ...  }
  virtual finance_ptr get(const char* indice)
  {
    int i = lookup(indice);
    return i<0 ? (finance_ptr)0
               : finance::_duplicate(objets[i]);
  }
  virtual void add(const char* indice, finance_ptr obj)
  {
    int i = lookup(indice);
    if (i<0) {
      i=nb; nb+=1;
      name[i] = strdup(indice);
    }
    objets[i] = obj->_duplicate(obj);
  }
};
```

Ex. finance (répartie)- Serveur GPS: Programme principal (fichier: gps.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;

  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...

  // mise sur l'orb d'un objet.
  gpsImpl* igps = new gpsImpl();

  // publication de l'ior de l'objet
  f = igps->_this();
  printf("%s\n", orb->object_to_string(f) );

  // lancement du serveur
  poa->the_POAManager()->activate();
  orb->run();

  return 0;
}
```

Ex. finance (répartie)- Serveur GPS: Programme principal (fichier: gps.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;

  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...

  // mise sur l'orb d'un objet.
  gpsImpl* igps = new gpsImpl();

  // publication de l'ior de l'objet
  f = igps->_this();
  printf("%s\n", orb->object_to_string(f) );

  // lancement du serveur
  poa->the_POAManager()->activate();
  orb->run();

  return 0;
}
```

Ex. finance (répartie)- Serveur GPS: Programme principal (fichier: gps.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;

  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...

  // mise sur l'orb d'un objet.
  gpsImpl* igps = new gpsImpl();

  // publication de l'ior de l'objet
  f = igps->_this();
  printf("%s\n", orb->object_to_string(f) );

  // lancement du serveur
  poa->the_POAManager()->activate();
  orb->run();

  return 0;
}
```

Ex. finance (répartie)- Serveur GPS: Programme principal (fichier: gps.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;

  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...

  // mise sur l'orb d'un objet.
  gpsImpl* igps = new gpsImpl();

  // publication de l'ior de l'objet
  f = igps->_this();
  printf("%s\n", orb->object_to_string(f) );

  // lancement du serveur
  poa->the_POAManager()->activate();
  orb->run();

  return 0;
}
```

Ex. finance (répartie)- Serveur GPS: Programme principal (fichier: gps.cc)

```
int main(int argc, char** argv)
{
  CORBA::Object* f;

  // initialisation de l'orb (serveur)
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa= ...

  // mise sur l'orb d'un objet.
  gpsImpl* igps = new gpsImpl();

  // publication de l'ior de l'objet
  f = igps->_this();
  printf("%s\n", orb->object_to_string(f) );

  // lancement du serveur
  poa->the_POAManager()->activate();
  orb->run();

  return 0;
}
```

Ex. finance (répartie)- Client C++ : Souche cliente de finance (fichier: finance.h généré)

```
class finance : virtual public CORBA::Object {
   ...
   CORBA::Long getcurr( CORBA::Double_out value );
   CORBA::Long getlast( ::DoubleArray_out values );
   ...
};
```

Ex. finance (répartie)- Client C++ : Programme principal (fichier: client.cc)

```
int main(int argc, char** argv) {
  CORBA::Object* f;double value;
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  // récupération d'un fantôme du GPS
  char* ior=argv[1];
  f = ...
  gps* fgps = ...

  // récupération d'un fantôme d'indice
  finance* findice =  ...

  // utilisation de findice (getcurr)
  if ( .....................<0 )
    printf("%s:getcurr: non disponible.\n");
  else
    printf("%s:getcurr:value = %6.1f\n",argv[2],value);

  return 0;
}
```

Ex. finance (répartie)- Client C++ : Programme principal (fichier: client.cc)

```
int main(int argc, char** argv) {
  CORBA::Object* f;double value;
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  // récupération d'un fantôme du GPS
  char* ior=argv[1];
  f = ...
  gps* fgps = ...

  // récupération d'un fantôme d'indice
  finance* findice =  ...

  // utilisation de findice (getcurr)
  if ( .....................<0 )
    printf("%s:getcurr: non disponible.\n");
  else
    printf("%s:getcurr:value = %6.1f\n",argv[2],value);

  return 0;
}
```

Ex. finance (répartie)- Client C++ : Programme principal (fichier: client.cc)

```
int main(int argc, char** argv) {
  CORBA::Object* f;double value;

  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  // récupération d'un fantôme du GPS
  char* ior=argv[1];
  f = ...
  gps* fgps = ...

  // récupération d'un fantôme d'indice
  finance* findice =  ...

  // utilisation de findice (getcurr)
  if ( .....................<0 )
    printf("%s:getcurr: non disponible.\n");
  else
    printf("%s:getcurr:value = %6.1f\n",argv[2],value);

  return 0;
}
```

Ex. finance (répartie)- Client C++ : Programme principal (fichier: client.cc)

```
int main(int argc, char** argv) {
  CORBA::Object* f;double value;

  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  // récupération d'un fantôme du GPS
  char* ior=argv[1];
  f = ...
  gps* fgps = ...

  // récupération d'un fantôme d'indice
  finance* findice =  ...

  // utilisation de findice (getcurr)
  if ( .....................<0 )
    printf("%s:getcurr: non disponible.\n");
  else
    printf("%s:getcurr:value = %6.1f\n",argv[2],value);

  return 0;
}
```

Ex. finance (répartie)- Client C++ : Programme principal (fichier: client.cc)

```
int main(int argc, char** argv) {
  CORBA::Object* f;double value;
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);

  // récupération d'un fantôme du GPS
  char* ior=argv[1];
  f = ...
  gps* fgps = ...

  // récupération d'un fantôme d'indice
  finance* findice =  ...

  // utilisation de findice (getcurr)
  if ( .....................<0 )
    printf("%s:getcurr: non disponible.\n");
  else
    printf("%s:getcurr:value = %6.1f\n",argv[2],value);

  return 0;
}
```

Ex. finance (répartie)- Client Java : Souches clientes

du GPS

```
public interface gpsOperations {
  finance get (String indice);
  void add (String indice, finance objet);
}
public interface gps extends gpsOperations, ... { }
```

de finance

```
public interface financeOperations
{
  int getcurr (org.omg.CORBA.DoubleHolder value);
  int getlast (DoubleArrayHolder values);
}
public interface finance extends financeOperations, ... { }
```

Ex. finance (répartie)- Client Java : Programme principal (fichier: client.java)

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);

    // récupération d'un fantôme du GPS
    String ior = args[0];
    org.omg.CORBA.Object f = ...
    gps fgps = ...

    // récupération d'un fantôme d'indice
    finance findice = ........................

    // utilisation de findice (getcurr)
    ..........................................
    if ( ..................................<0 )
      System.out.println(args[1] + " : non disponible");
    else
      System.out.println(args[1] + " : " + ..................
 }
}
```

Ex. finance (répartie)- Client Java : Programme principal (fichier: client.java)

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);
    // récupération d'un fantôme du GPS
    String ior = args[0];
    org.omg.CORBA.Object f = ...
    gps fgps = ...
    // récupération d'un fantôme d'indice
    finance findice = ..........................
    // utilisation de findice (getcurr)
    ..............................................
    if ( ..................................<0 )
      System.out.println(args[1] + " : non disponible");
    else
      System.out.println(args[1] + " : " + ..................
 }
}
```

Ex. finance (répartie)- Client Java : Programme principal (fichier: client.java)

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);
    // récupération d'un fantôme du GPS
    String ior = args[0];
    org.omg.CORBA.Object f = ...
    gps fgps = ...
    // récupération d'un fantôme d'indice
    finance findice = ..........................
    // utilisation de findice (getcurr)
    ..............................................
    if ( ..................................<0 )
      System.out.println(args[1] + " : non disponible");
    else
      System.out.println(args[1] + " : " + ..................
 }
}
```

Ex. finance (répartie)- Client Java : Programme principal (fichier: client.java)

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);
    // récupération d'un fantôme du GPS
    String ior = args[0];
    org.omg.CORBA.Object f = ...
    gps fgps = ...
    // récupération d'un fantôme d'indice
    finance findice = ..........................
    // utilisation de findice (getcurr)
    ..............................................
    if ( ..................................<0 )
      System.out.println(args[1] + " : non disponible");
    else
      System.out.println(args[1] + " : " + ..................
 }
}
```

Ex. finance (répartie)- Client Java : Programme principal (fichier: client.java)

```
public class client {
public static void main(String args[]) {
    ORB orb = ORB.init(args, null);

    // récupération d'un fantôme du GPS
    String ior = args[0];
    org.omg.CORBA.Object f = ...
    gps fgps = ...

    // récupération d'un fantôme d'indice
    finance findice = ........................

    // utilisation de findice (getcurr)
    ..........................................
    if ( ..................................<0 )
      System.out.println(args[1] + " : non disponible");
    else
      System.out.println(args[1] + " : " + ..................
 }
}
```

Table des matières

Obtention des IORs

Quelles IORs

```
interface X {  | interface Y {
   ...         |    ...
   ...         |    X getX();
   ...         |    ...
}              | }
```

Seules les racines des graphes d'objets sont nécessaires.

Méthodes

- cablé en dur
- par fichiers
- par un autre protocole http, ldap, ...
- Naming Service: serveur CORBA stockant des couples (nom,IOR).
- Trader Service: serveur CORBA stockant des couples (mot clé, IOR).

Obtention des IORs

Quelles IORs

```
interface X {  |  interface Y {
   ...         |     ...
   ...         |     X getX();
   ...         |     ...
}              |  }
```

Seules les racines des graphes d'objets sont nécessaires.

Méthodes

- cablé en dur
- par fichiers
- par un autre protocole http, ldap, ...
- Naming Service: serveur CORBA stockant des couples (nom,IOR).
- Trader Service: serveur CORBA stockant des couples (mot clé, IOR).

Gestion des références

Langage OO

Java $\Longrightarrow$ c'est fait tout seul

C++ $\Longrightarrow$ à la charge du programmeur

CORBA

Java $\Longrightarrow$ c'est fait tout seul

C++ $\Longrightarrow$ CORBA ajoute un compteur de references dans les objets réels et *fantômes*.

Middleware OO

Quelque soit le langage, quand un serveur sert un objet il ne peut être détruit.

$\Longrightarrow$ le client doit indiquer qu'il n'en a plus besoin

$\Longrightarrow$ nécessite un nettoyage pèriodique pour les clients mal programmé ou pour les crashs

Actions périodiques sur un serveur

Problème

```
CORBA::ORB_var           orb;
PortableServer::POA_var poa;
...
poa->the_POAManager()->activate();
orb->run();
```

Solution système

- Exemple UNIX: faire un callback sur un des signaux SIGUSR$_i$, SIGHUP.
- Exemple UNIX: faire un callback sur SIGALRM et programmer le timer.

$\longrightarrow$ non portable

Solution middleware

Ajouter un objet avec une fonction "do()" dans le serveur, lancer un client periodiquement (crond) qui fait "do".

Actions périodiques sur un serveur

Problème

```
orb->run();
```

La boucle infinie d'écoute sur l'ORB empêche la possibilité de toute action.

Solution système

- Exemple UNIX: faire un callback sur un des signaux $SIGUSR_i$, SIGHUP.
- Exemple UNIX: faire un callback sur SIGALRM et programmer le timer.

$\longrightarrow$ non portable

Solution middleware

Ajouter un objet avec une fonction "do()" dans le serveur, lancer un client periodiquement (crond) qui fait "do".

Actions périodiques sur un serveur

Problème

```
orb->run();
```

La boucle infinie d'écoute sur l'ORB empêche la possibilité de toute action.

Solution système

- Exemple UNIX: faire un callback sur un des signaux SIGUSR$_i$, SIGHUP.
- Exemple UNIX: faire un callback sur SIGALRM et programmer le timer.

$\longrightarrow$ non portable

Solution middleware

Ajouter un objet avec une fonction "do()" dans le serveur, lancer un client periodiquement (crond) qui fait "do".

Exemple d'implantation simple d'un serveur

Principe

Le processus est soit client soit serveur.

Algorithme

pour toujours faire
  PDUR = lire-reseau(client)
  Obj,Meth,args $\Longleftarrow$ PDUR
  ret $\Longleftarrow$ Obj$->$Meth(args)
  PDUA $\Longleftarrow$ OK,ret
  ecrire-reseau(PDUA,client)
fait

Exemple d'implantation simple d'un serveur

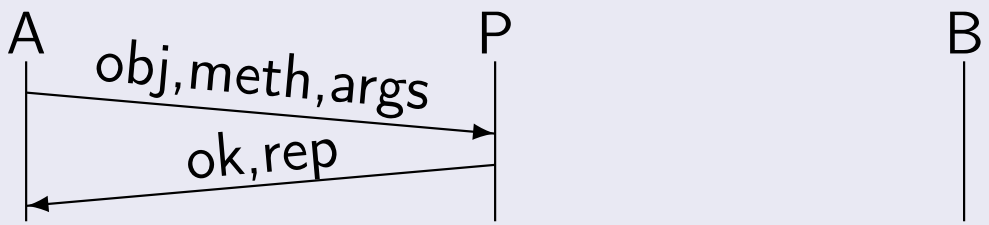
Principe

Le processus est soit client soit serveur.

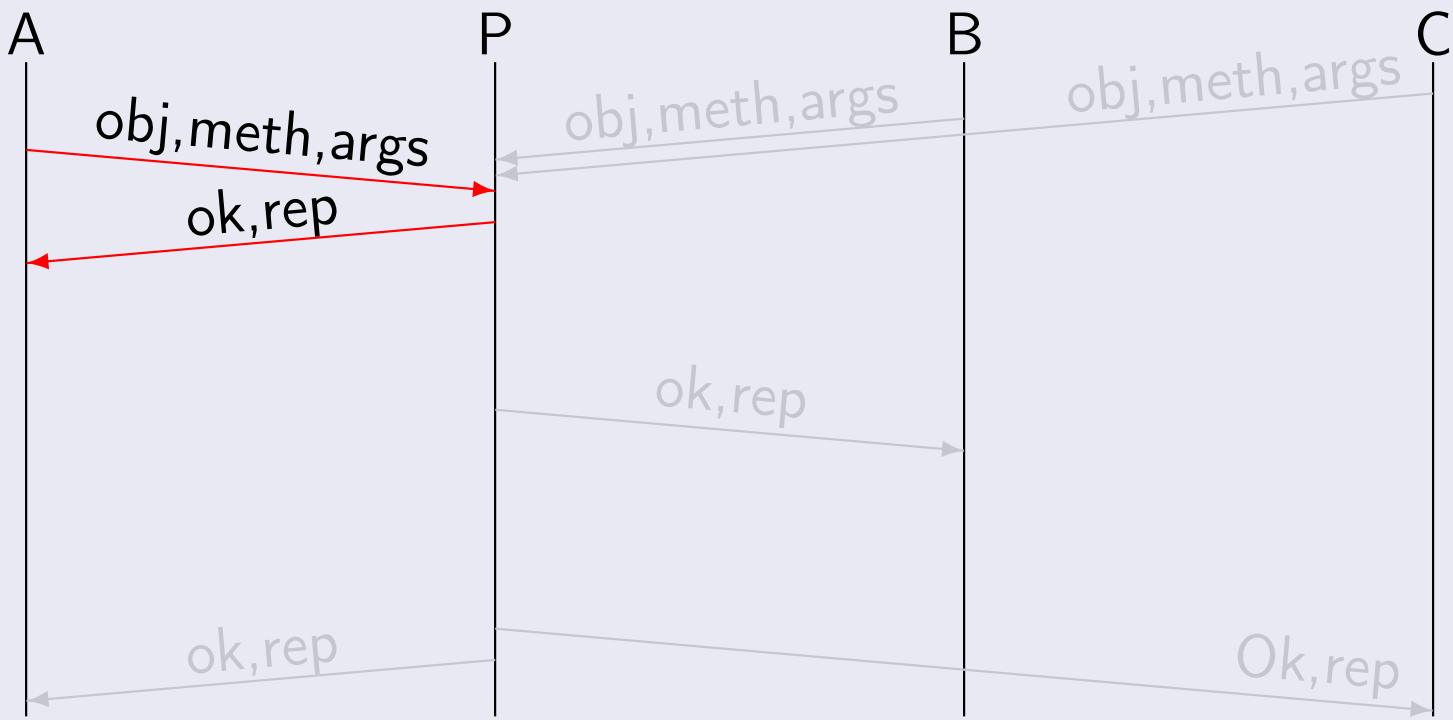
Algorithme

pour toujours faire
  PDUR = lire-reseau(client)
  Obj,Meth,args $\Longleftarrow$ PDUR
  ret $\Longleftarrow$ Obj$->$Meth(args)
  PDUA $\Longleftarrow$ OK,ret
  ecrire-reseau(PDUA,client)
fait

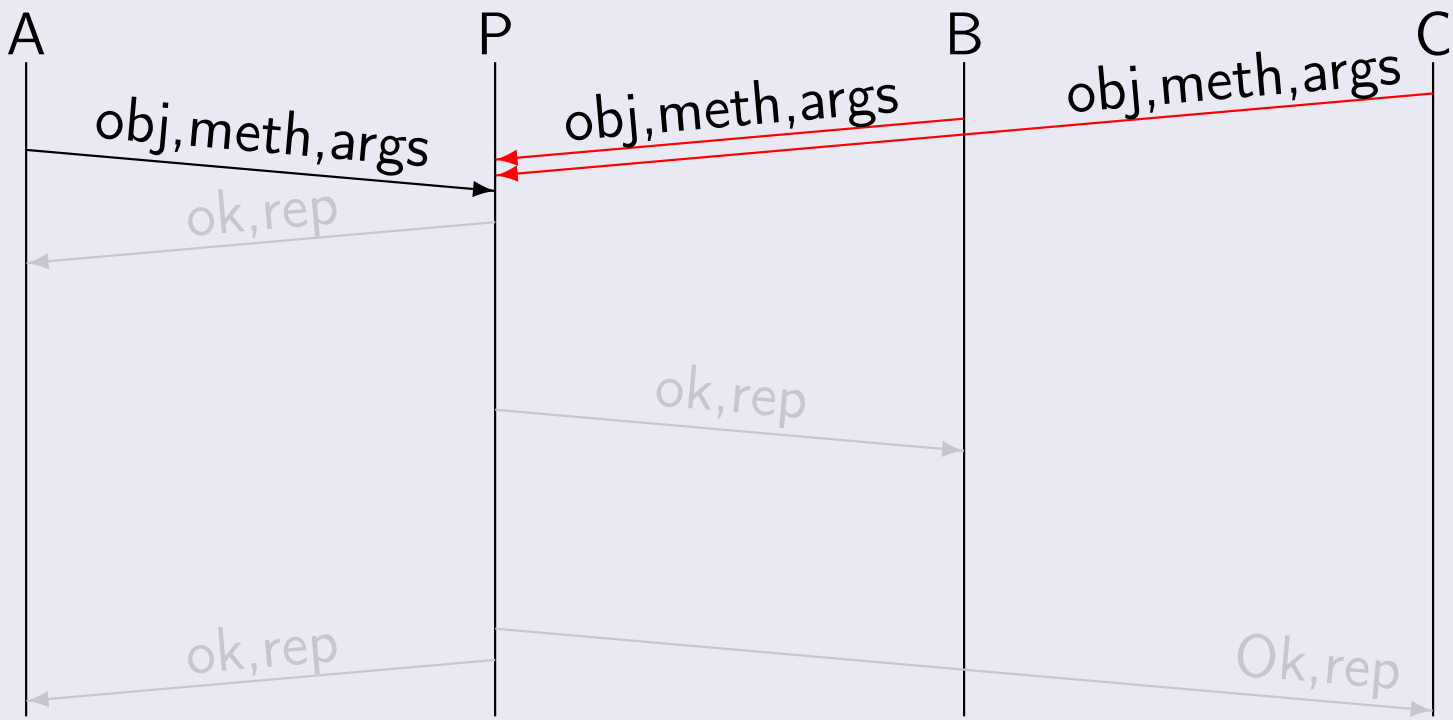
Exemple 1: $A \longrightarrow P$



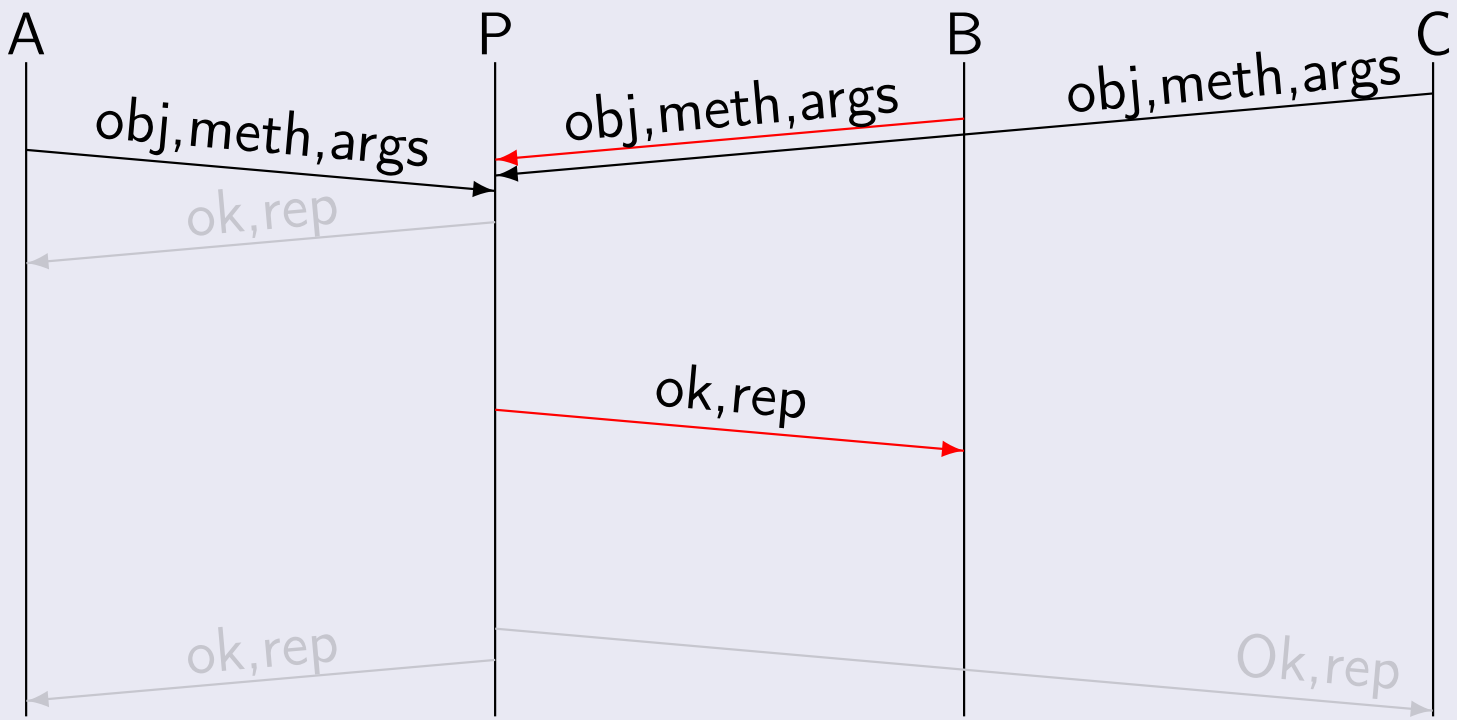
Exemple 2: $B \rightarrow P$, $C \rightarrow P$, $A \rightarrow P$



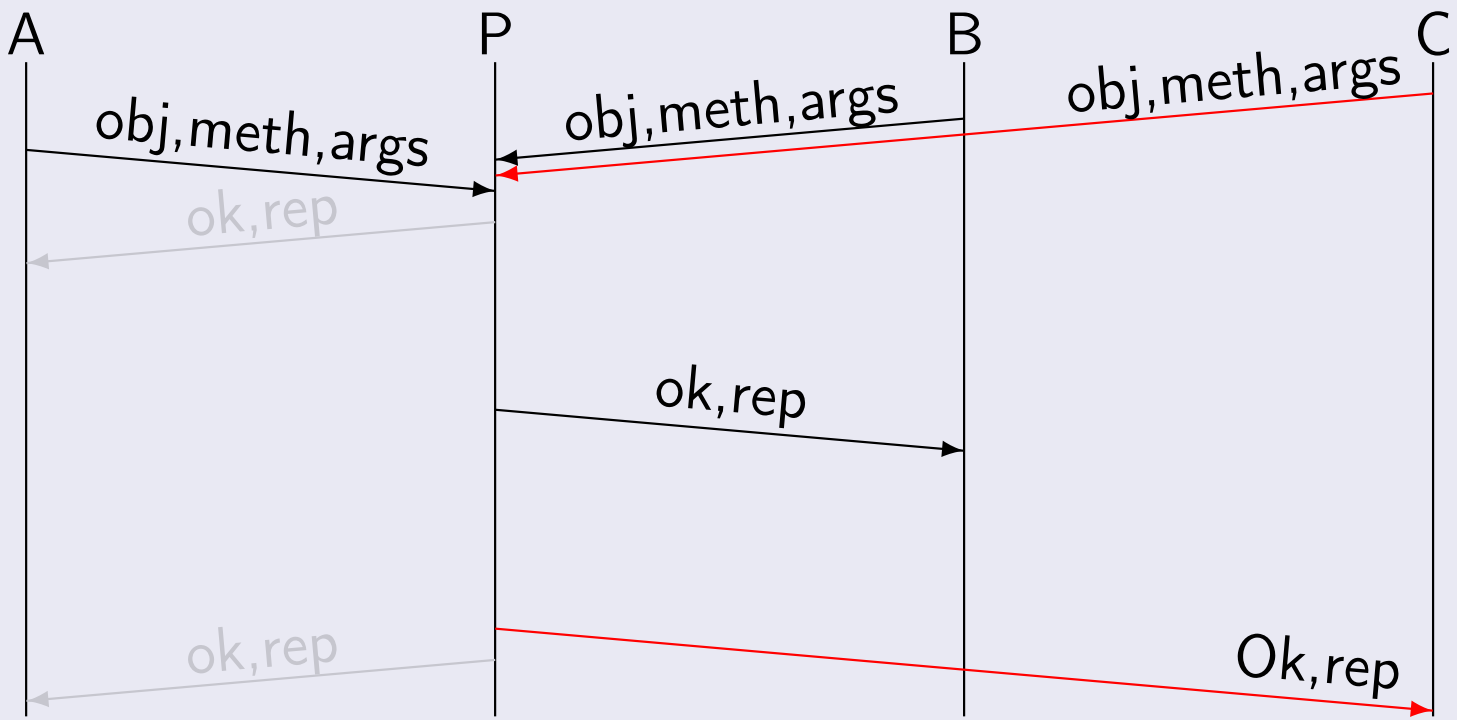
Exemple 2: $B \rightarrow P$, $C \rightarrow P$, $A \rightarrow P$



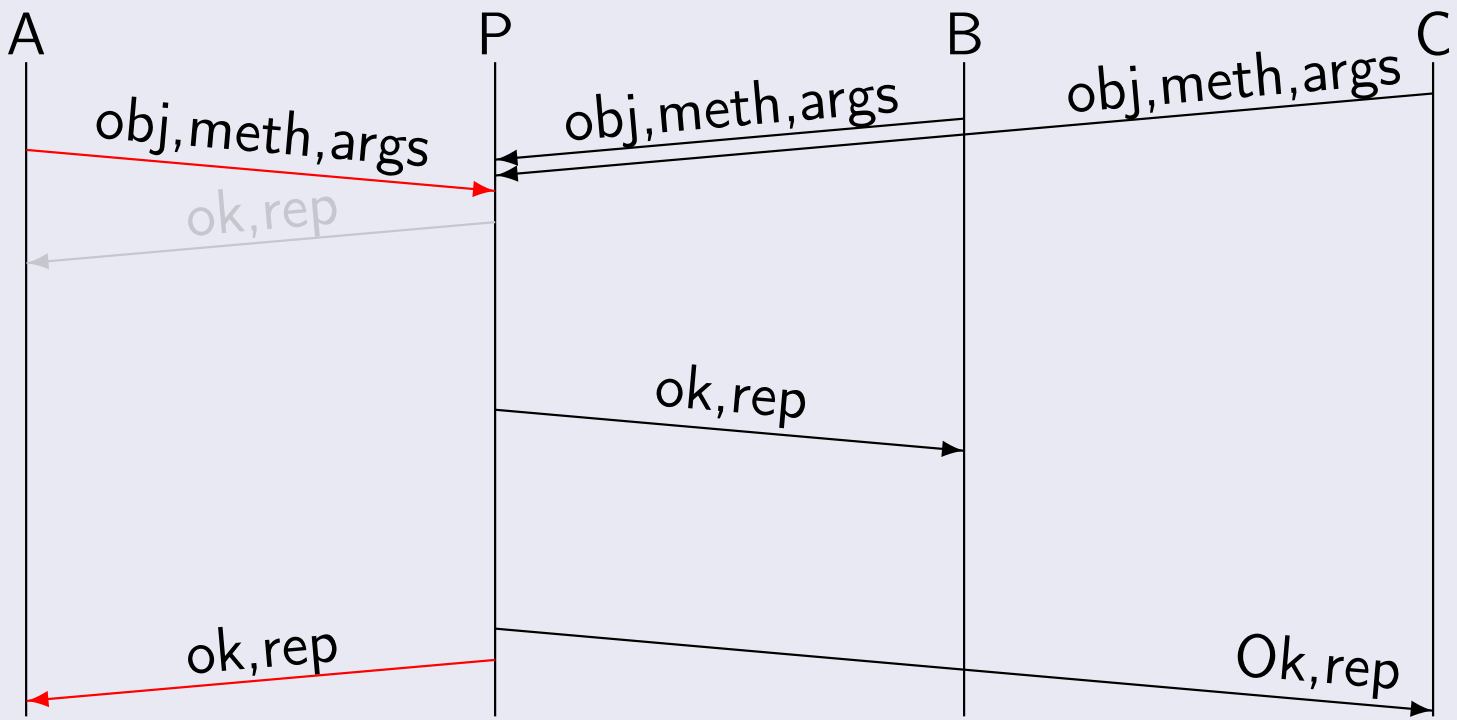
Exemple 2: $B \rightarrow P$, $C \rightarrow P$, $A \rightarrow P$



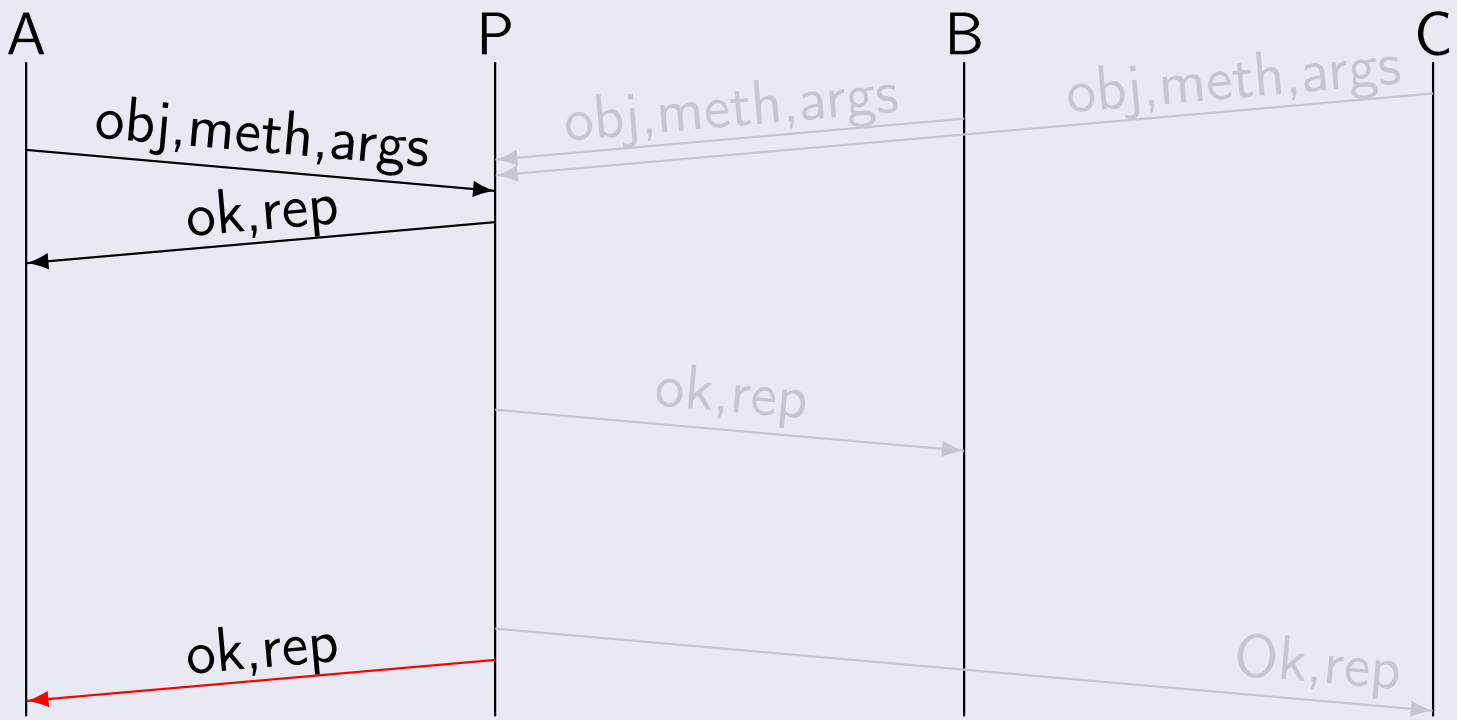
Exemple 2: $B \rightarrow P$, $C \rightarrow P$, $A \rightarrow P$



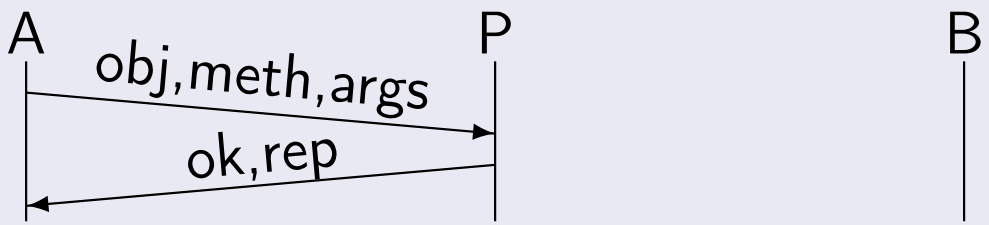
Exemple 2: $B \rightarrow P$, $C \rightarrow P$, $A \rightarrow P$



Exemple 2: $B \rightarrow P$, $C \rightarrow P$, $A \rightarrow P$



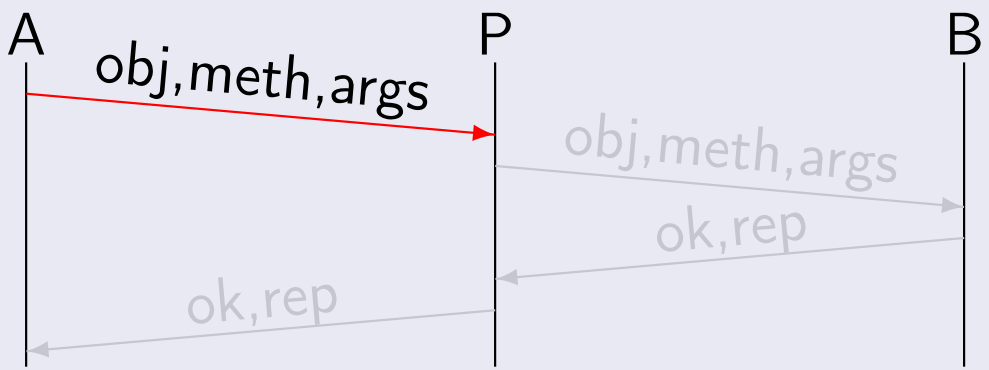
Exemple 1: $A \longrightarrow P$



blocage

- P attend une réponse sur la socket du serveur B.
- B attend une réponse sur la socket du serveur P.

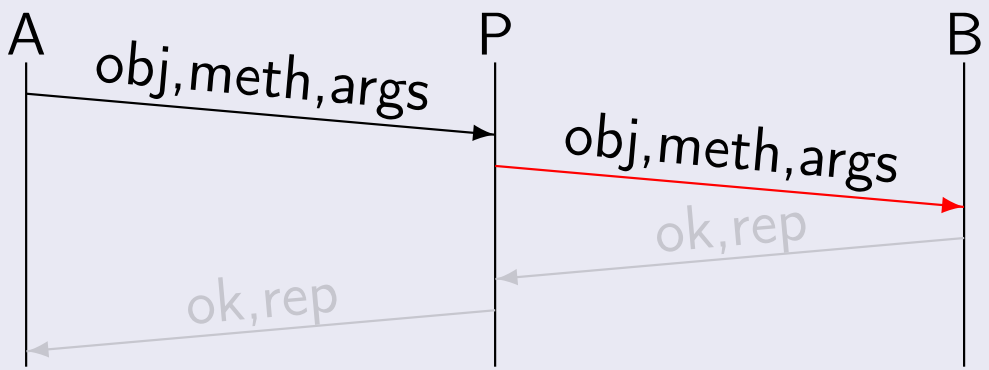
Exemple 2: $A \rightarrow P \rightarrow B$



blocage

- P attend une réponse sur la socket du serveur B.
- B attend une réponse sur la socket du serveur P.

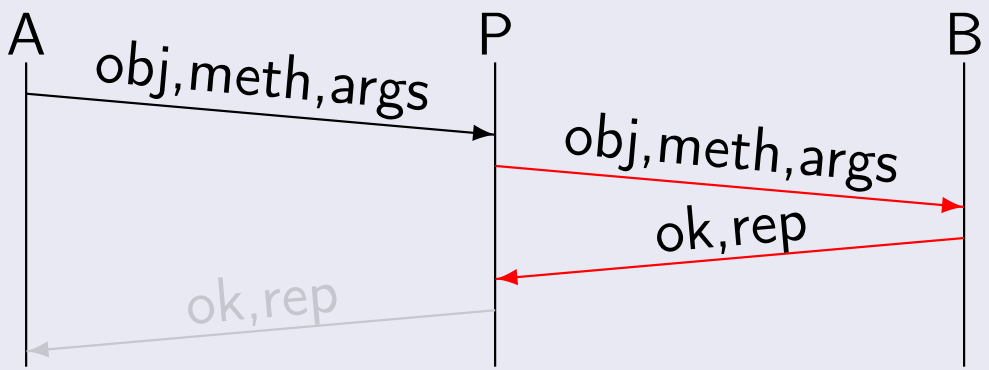
Exemple 2: $A \rightarrow P \rightarrow B$



blocage

- P attend une réponse sur la socket du serveur B.
- B attend une réponse sur la socket du serveur P.

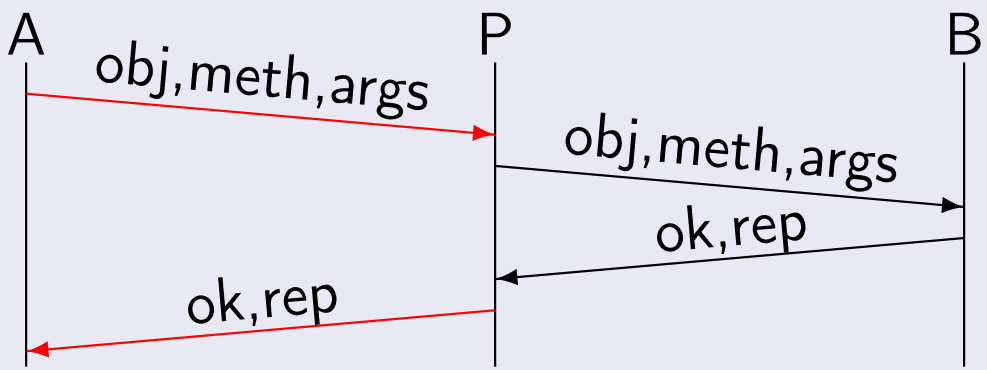
Exemple 2: $A \rightarrow P \rightarrow B$



blocage

- P attend une réponse sur la socket du serveur B.
- B attend une réponse sur la socket du serveur P.

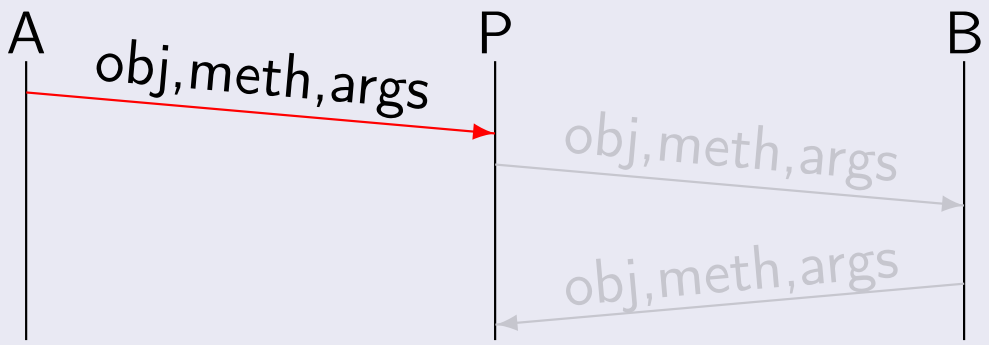
Exemple 2: $A \rightarrow P \rightarrow B$



blocage

- P attend une réponse sur la socket du serveur B.
- B attend une réponse sur la socket du serveur P.

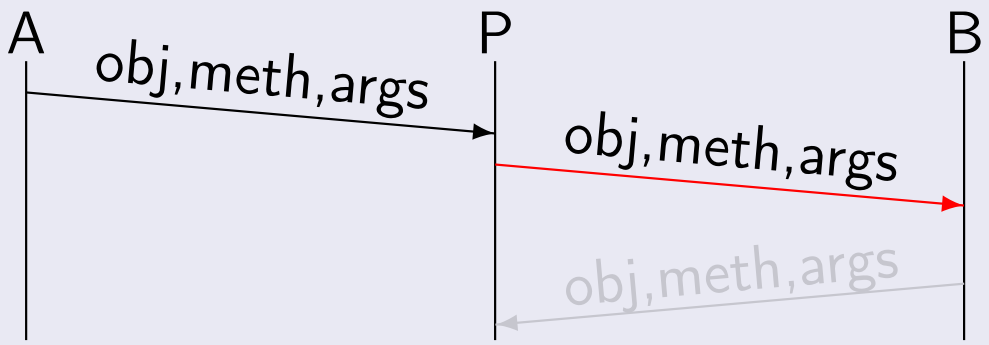
Exemple 3: $A \rightarrow P \rightarrow B \rightarrow P$



blocage

- P attend une réponse sur la socket du serveur B.
- B attend une réponse sur la socket du serveur P.

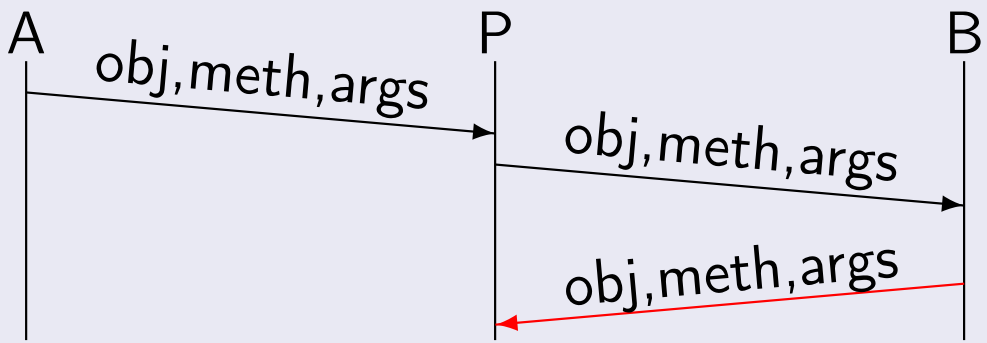
Exemple 3: $A \rightarrow P \rightarrow B \rightarrow P$



blocage

- P attend une réponse sur la socket du serveur B.
- B attend une réponse sur la socket du serveur P.

Exemple 3: $A \rightarrow P \rightarrow B \rightarrow P$



blocage

- P attend une réponse sur la socket du serveur B.
- B attend une réponse sur la socket du serveur P.

Solution réseau classique (V1)

- Créer une thread pour chaque requête sortante.
- Créer une thread pour chaque requête entrante.

Solution réseau classique (V1)

- Créer une thread pour chaque requête sortante.
- Créer une thread pour chaque requête entrante.

Avantages

- Résout les problemes d'inter-blocage.
- Résout partiellement les problemes de réactivité.

Inconvénients

- Ralentit les fonctions simples.
  $\Longrightarrow$ Réactivité constante mais dégradée.
- Grosse charge
  $\Longrightarrow$ Risque d'écroulement de la machine.

Solution réseau classique (V2)

Créer un pool de threads.

- prendre une thread du pool pour chaque requête sortante.
- prendre une thread du pool pour chaque requête entrante.
- si pas de thread libre attendre qu'une se libère.

Solution réseau classique (V2)

Créer un pool de threads.

- prendre une thread du pool pour chaque requête sortante.
- prendre une thread du pool pour chaque requête entrante.
- si pas de thread libre attendre qu'une se libère.

Avantages

- Répousse les problemes d'inter-blocage.
- Résout les problemes de réactivité.
- Fixe une borne au ressources système utilisées.

Inconvénients

- Inter-blocages encore possibles.
- Quitte le monde séquentiel.

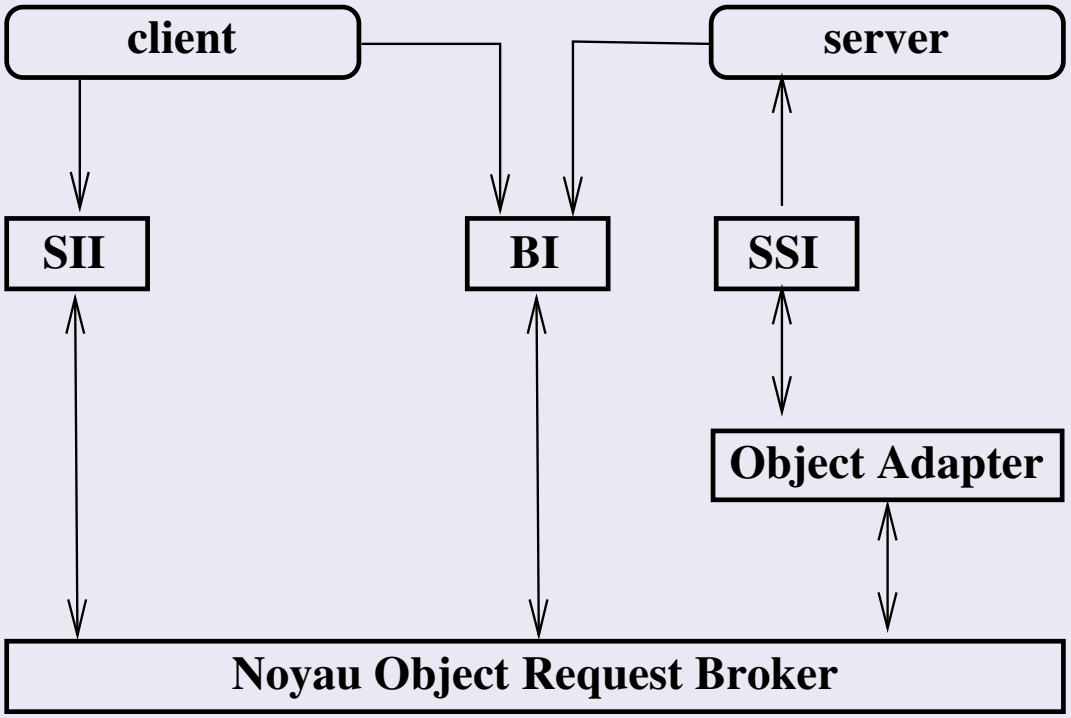
Solution CORBA

- Possibilité de créer des POAs (threads serveurs).
- Possibilité d'associer chaque objet à un POA.
- Pour des objets peu utilisés
  - Possibilité de désactiver un objet quand il n'est pas utilisé.
  - L'objet est réactivité automatiquement quand une requête arrive.

$\Longrightarrow$ libère des ressources système.

$\Longrightarrow$ permet de réaliser des serveurs avec beaucoup d'objets.

Architecture



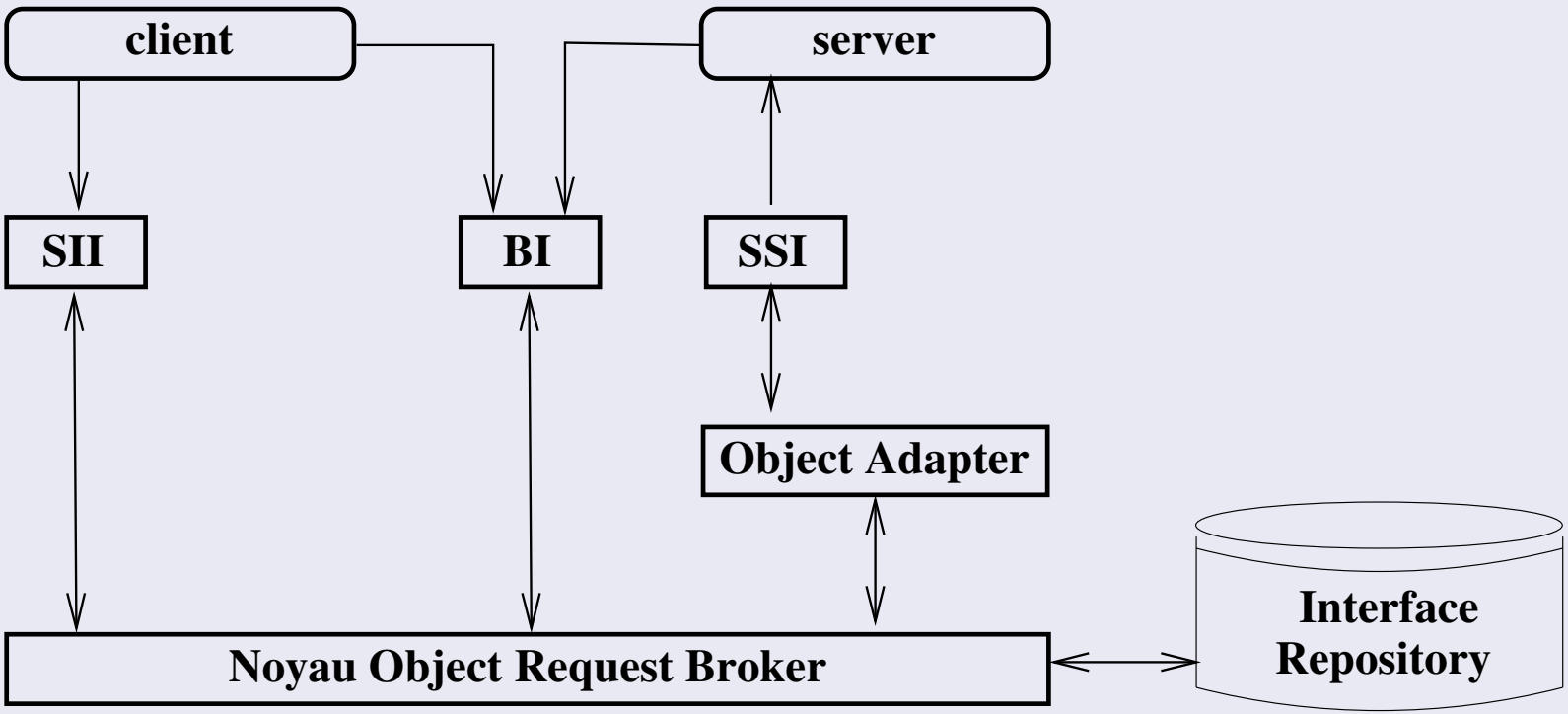
ORB Noyau de communication (Object Request Broker)

BI/OA Bus Interface / Adadapteur d'objet

SII Static Invocation Interface

SSI Static Squeleton Interface

Architecture



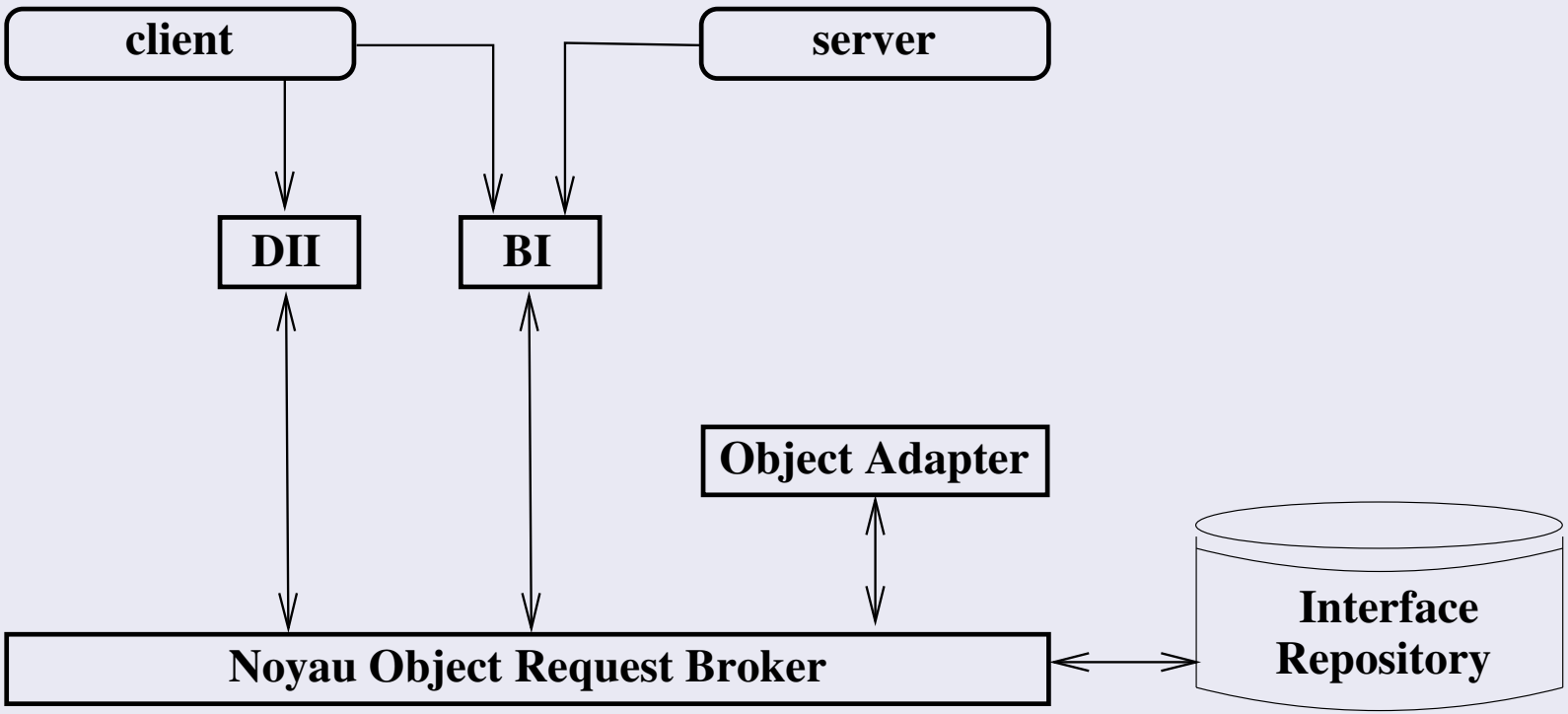
I. R. Description des interfaces des objets

$\Longrightarrow$ introspection

DII Dynamic Invocation Interface

DII Dynamic Squeleton Interface

Architecture

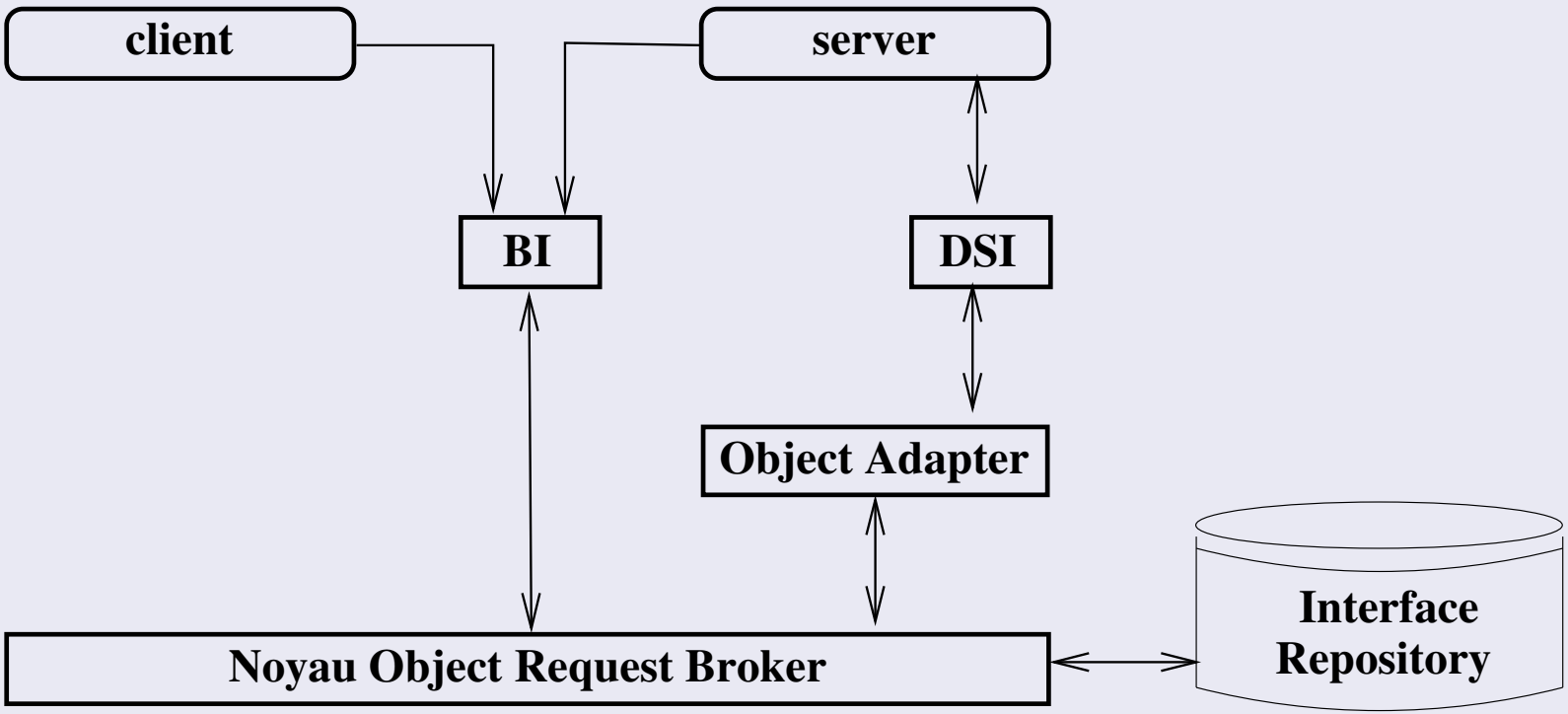


I. R. Description des interfaces des objets
$\Longrightarrow$ introspection

DII Dynamic Invocation Interface

DII Dynamic Squeleton Interface

Architecture

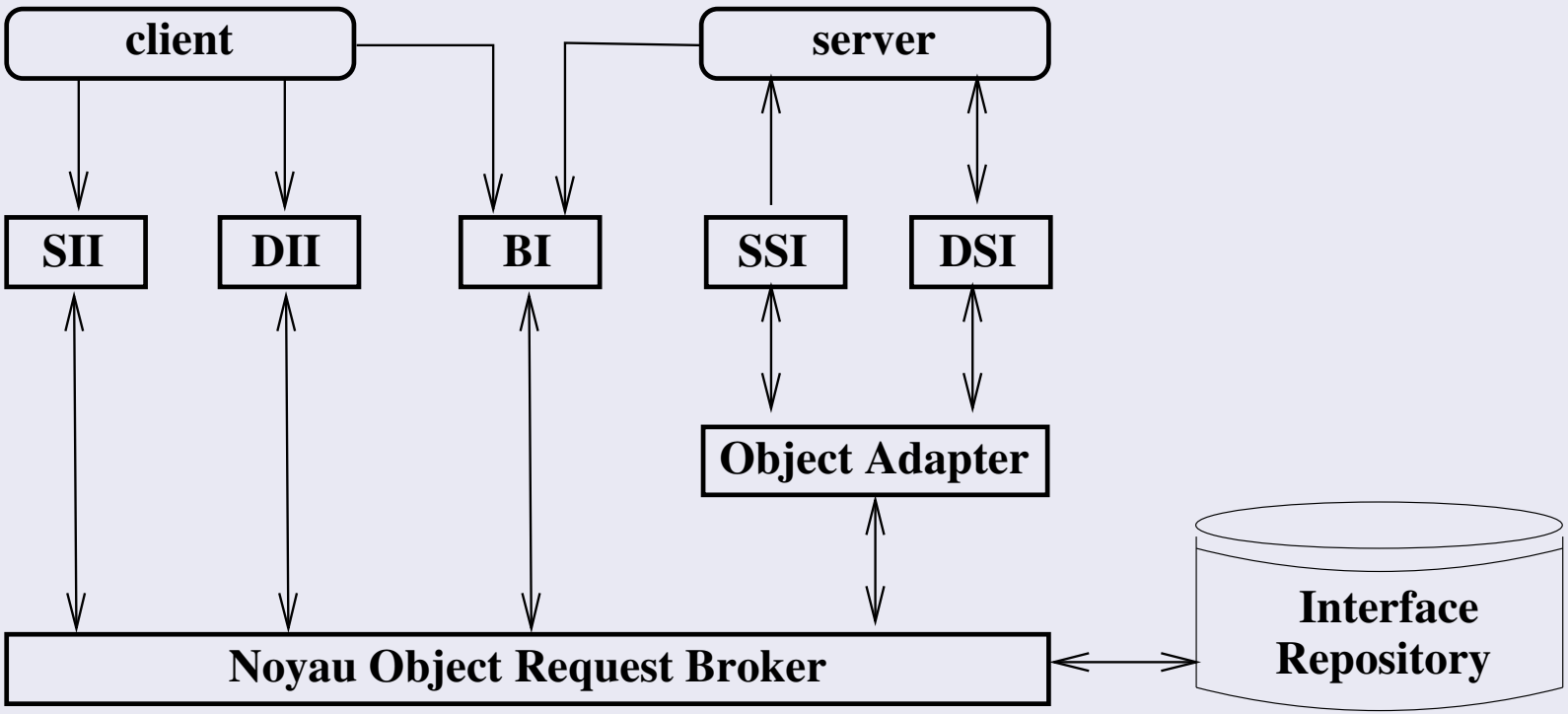


I. R. Description des interfaces des objets
$\Longrightarrow$ introspection

DII Dynamic Invocation Interface

DII Dynamic Squeleton Interface

Architecture



I. R. Description des interfaces des objets
$\Longrightarrow$ introspection

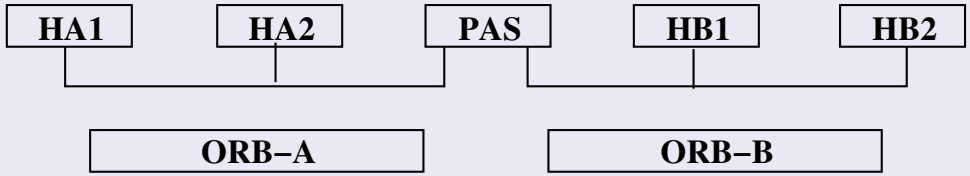
DII Dynamic Invocation Interface

DII Dynamic Squeleton Interface

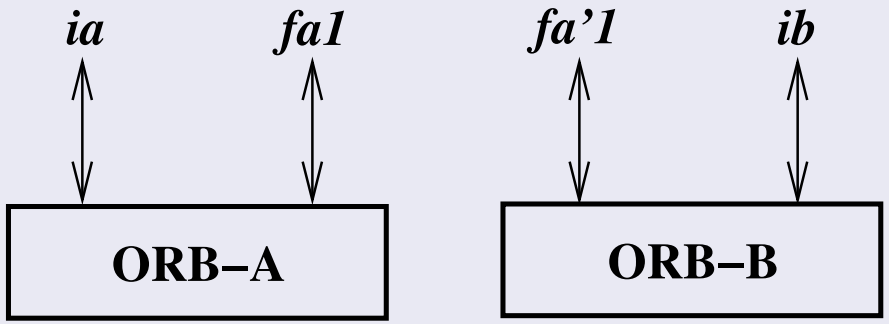
Ponts

utilité

- Explorateur du graphe d'objet (limité).
- Pont entre 2 ORBs.



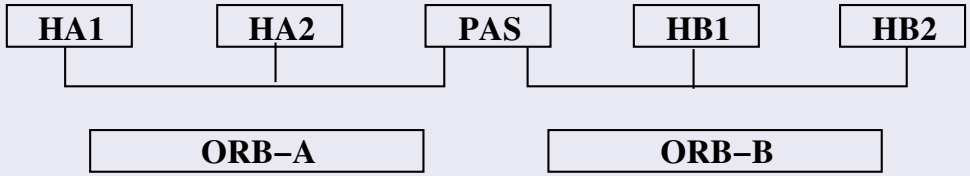
implantation



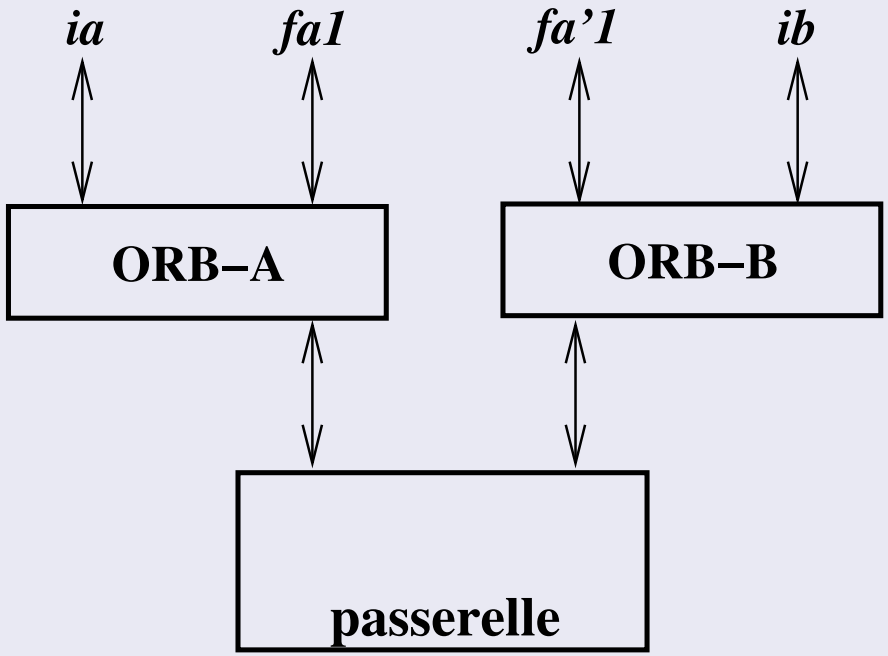
Ponts

utilité

- Explorateur du graphe d'objet (limité).
- Pont entre 2 ORBs.



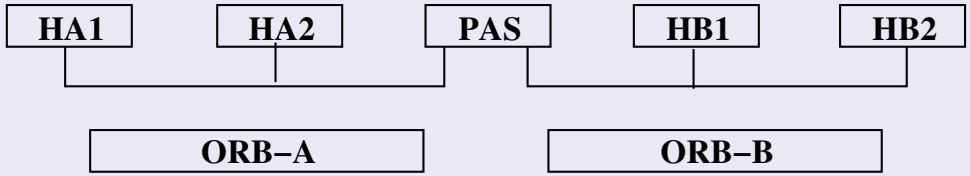
implantation



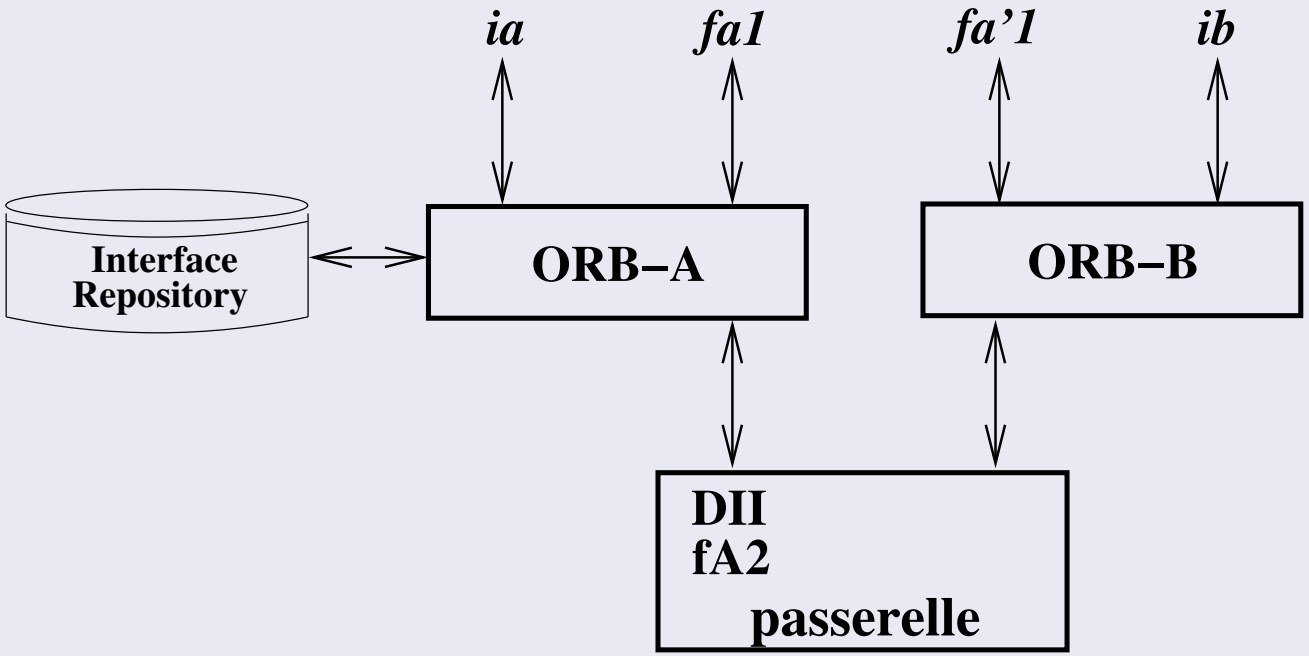
Ponts

utilité

- Explorateur du graphe d'objet (limité).
- Pont entre 2 ORBs.



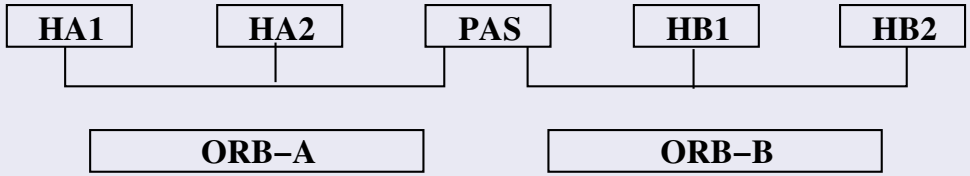
implantation



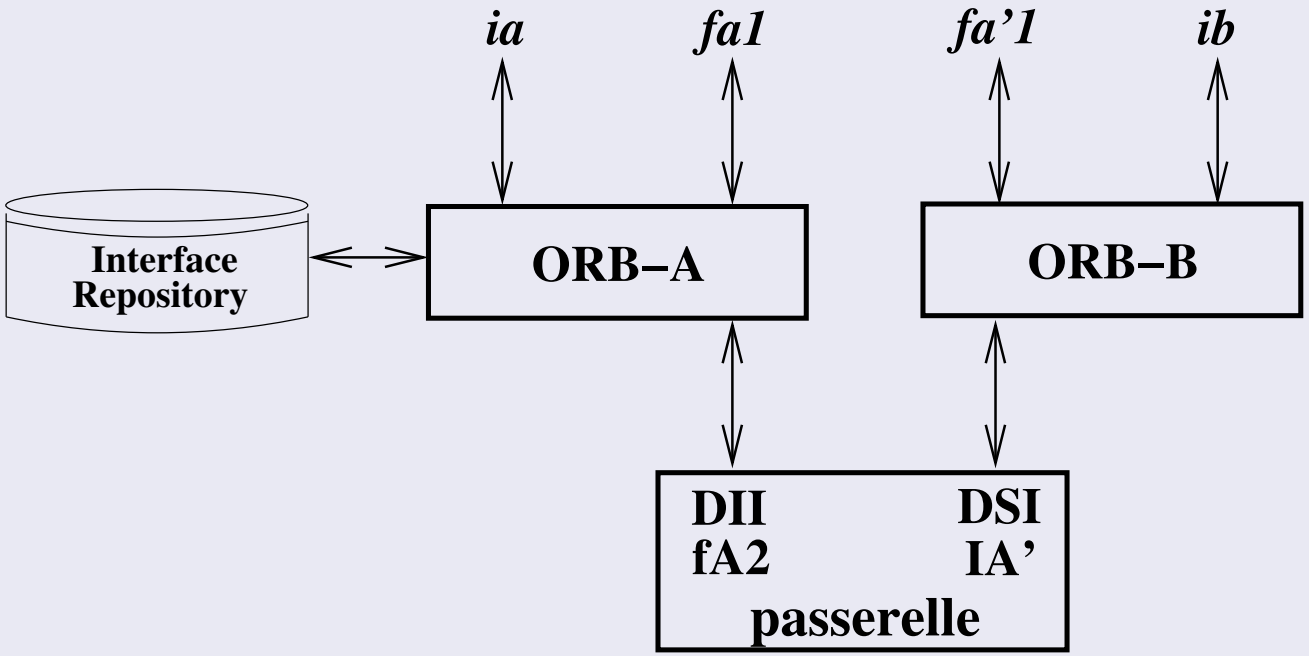
Ponts

utilité

- Explorateur du graphe d'objet (limité).
- Pont entre 2 ORBs.

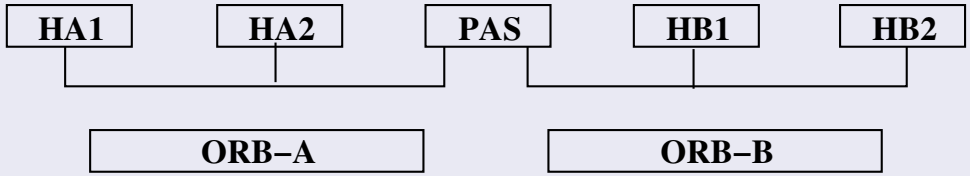


implantation

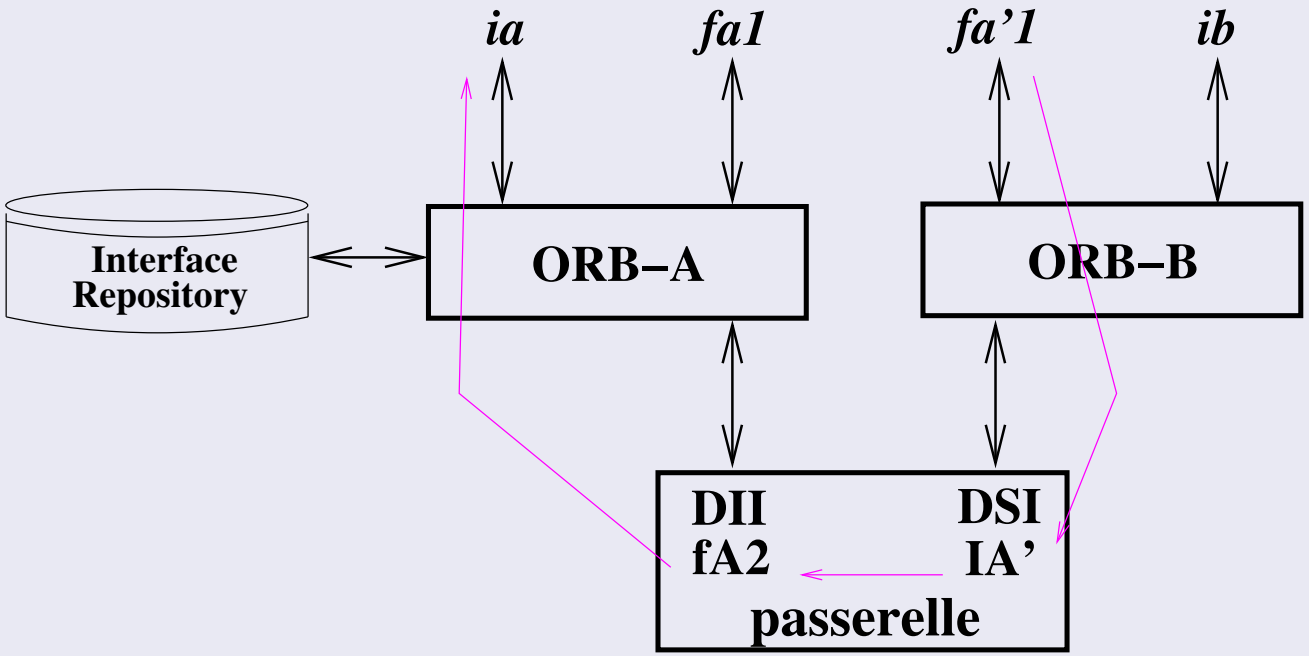


utilité

- Explorateur du graphe d'objet (limité).
- Pont entre 2 ORBs.

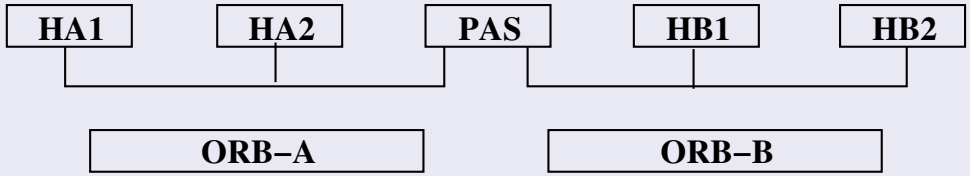


implantation



utilité

- Explorateur du graphe d'objet (limité).
- Pont entre 2 ORBs.



implantation

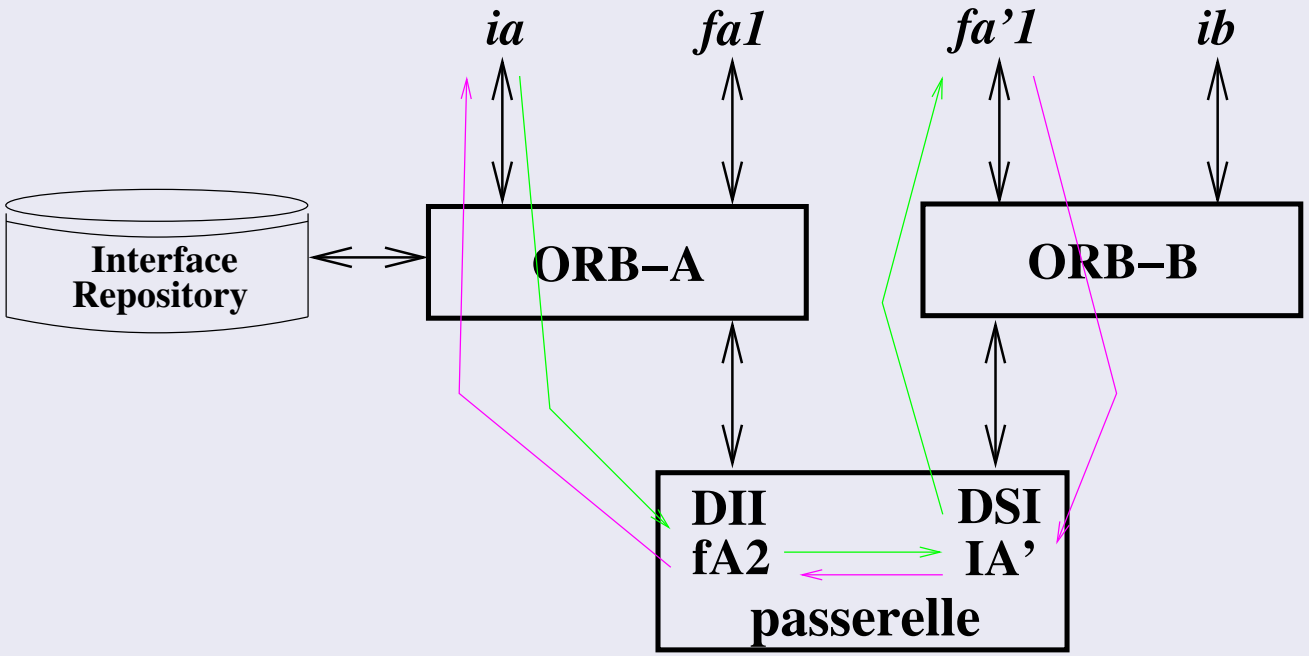


Table des matières

Principaux services

Collection, query gestion d'ensembles d'objet.

Concurrency accès en section critique à un objet.

Time synchronisation d'horloge.

EVoT enhanced view of time: Time + exécution périodique et diférée et batch.

Event, notification canaux de communication, signalisation.

Externalization persistence et internationnalisation.

Persistent state base de données d'objet.

Licensing gestionnaire de license.

naming associer un nom à un objet.

property associer des informations à des objets.

trading associer des mots clé à un objet.

transaction moteur transactionnel.

Spécification

Description idl Les services sont des interfaces CORBA.

Documentation utilisateur des interfaces utilisateurs.

$\Longrightarrow$ Chaque service est un ou un ensemble d'objets CORBA.

$\Longrightarrow$ Les implantations propriétaires CORBA fournissent:

- En librairie les souches clientes de ces interfaces.
- En librairie les squelette serveur de ces interfaces.
- Les exécutables de serveurs implantant ces interfaces.

Spécification

Description idl Les services sont des interfaces CORBA.

Documentation utilisateur des interfaces utilisateurs.

$\Longrightarrow$ Chaque service est un ou un ensemble d'objets CORBA.

$\Longrightarrow$ Les implantations propriétaires CORBA fournissent:

- En librairie les souches clientes de ces interfaces.
- En librairie les squelette serveur de ces interfaces.
- Les exécutables de serveurs implantant ces interfaces.

Gabarit

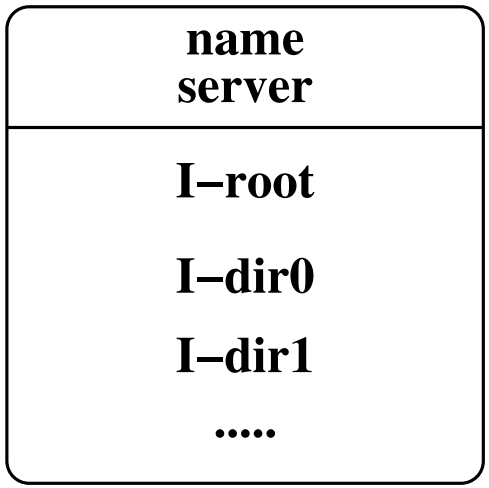
```
module ServiceName {
    /* description des types internes */
    ...
    /* definition des enumeration */
    ...
    /* definition des exceptions */
    ...
    /* definition des interfaces  */
    ...
};
```

NS: Introduction

Principe

1. Un serveur gérant des couples (nom,IOR).
2. Les objets serveur s'enregistrent dans le NS sous un nom.
3. Les clients récupèrent du NS un objet en fonction du nom.

$\Longrightarrow$ NS est la racine du graphe d'objets.



Structure des noms

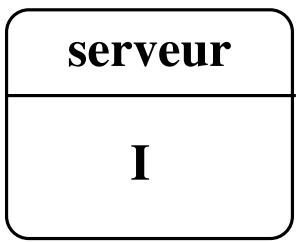
- structure arborescente (ex: $/rep_0/rep_1/.../nom$)
- nom de base: un couple (id.kind)

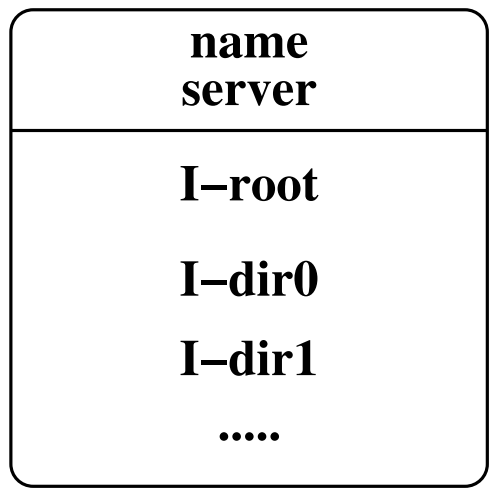
NS: Introduction

Principe

1. Un serveur gérant des couples (nom,IOR).
2. Les objets serveur s'enregistrent dans le NS sous un nom.
3. Les clients récupèrent du NS un objet en fonction du nom.

$\Longrightarrow$ NS est la racine du graphe d'objets.





Structure des noms

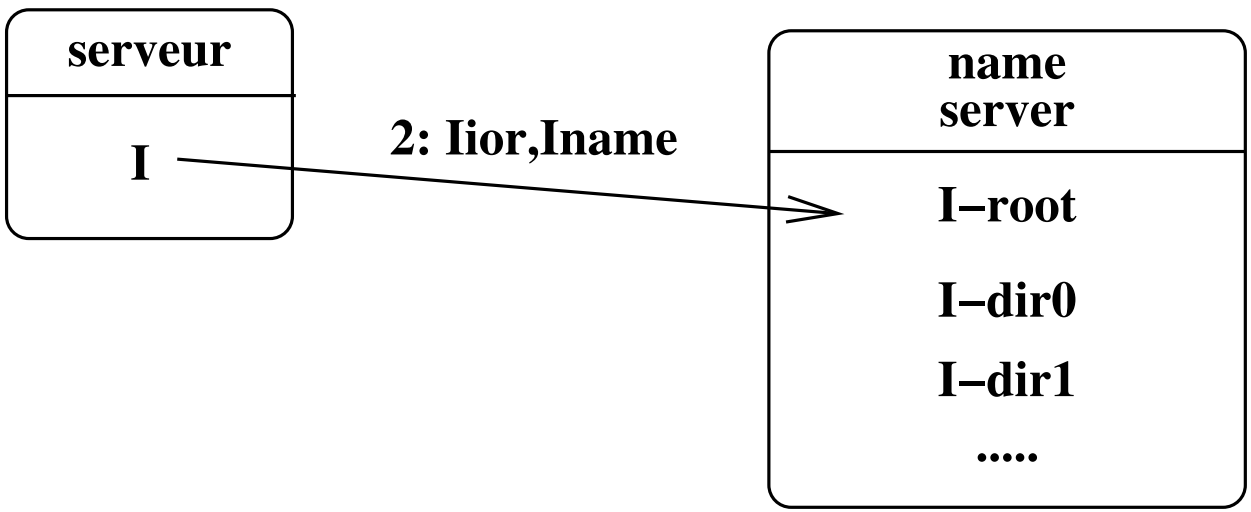
- structure arborescente (ex: $/rep_0/rep_1/.../nom$)
- nom de base: un couple (id.kind)

NS: Introduction

Principe

1. Un serveur gérant des couples (nom,IOR).
2. Les objets serveur s'enregistrent dans le NS sous un nom.
3. Les clients récupèrent du NS un objet en fonction du nom.

$\Longrightarrow$ NS est la racine du graphe d'objets.



Structure des noms

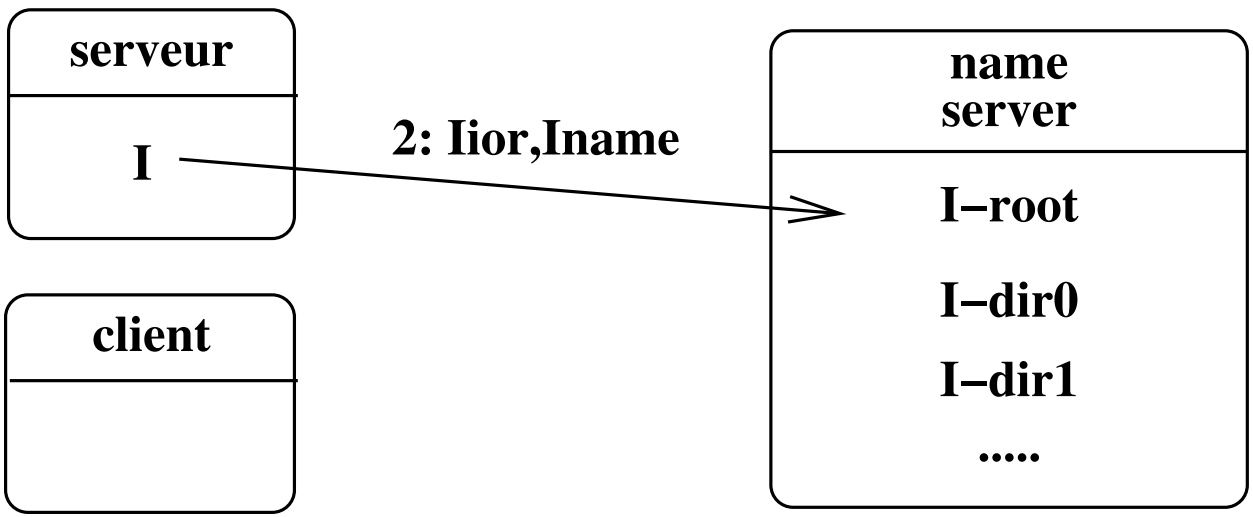
- structure arborescente (ex: $/rep_0/rep_1/.../nom$)
- nom de base: un couple (id.kind)

NS: Introduction

Principe

1. Un serveur gérant des couples (nom,IOR).
2. Les objets serveur s'enregistrent dans le NS sous un nom.
3. Les clients récupèrent du NS un objet en fonction du nom.

$\Longrightarrow$ NS est la racine du graphe d'objets.



Structure des noms

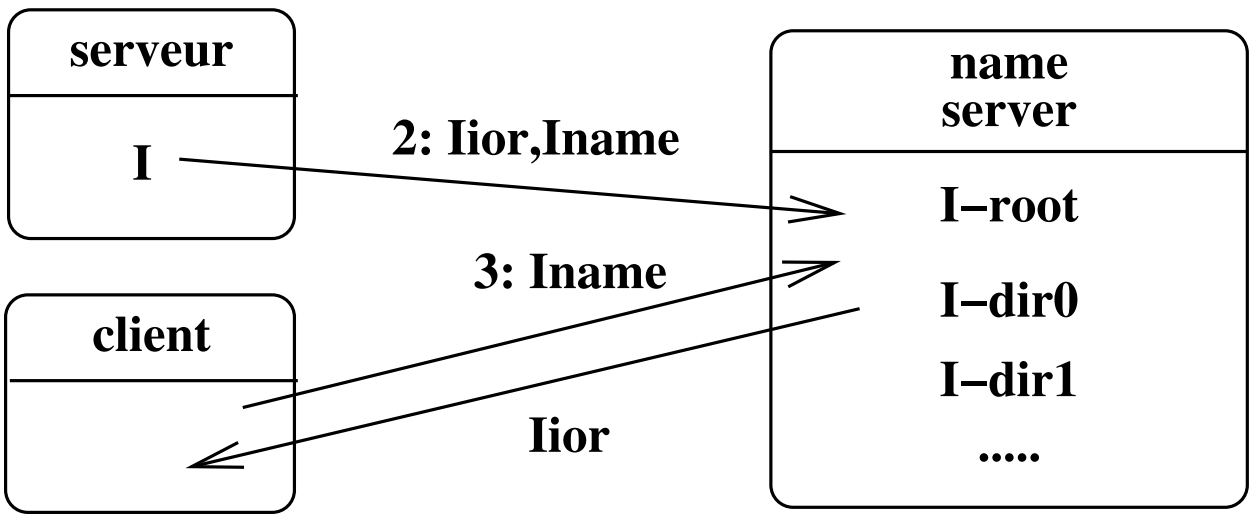
- structure arborescente (ex: $/rep_0/rep_1/.../nom$)
- nom de base: un couple (id.kind)

NS: Introduction

Principe

1. Un serveur gérant des couples (nom,IOR).
2. Les objets serveur s'enregistrent dans le NS sous un nom.
3. Les clients récupèrent du NS un objet en fonction du nom.

$\Longrightarrow$ NS est la racine du graphe d'objets.



Structure des noms

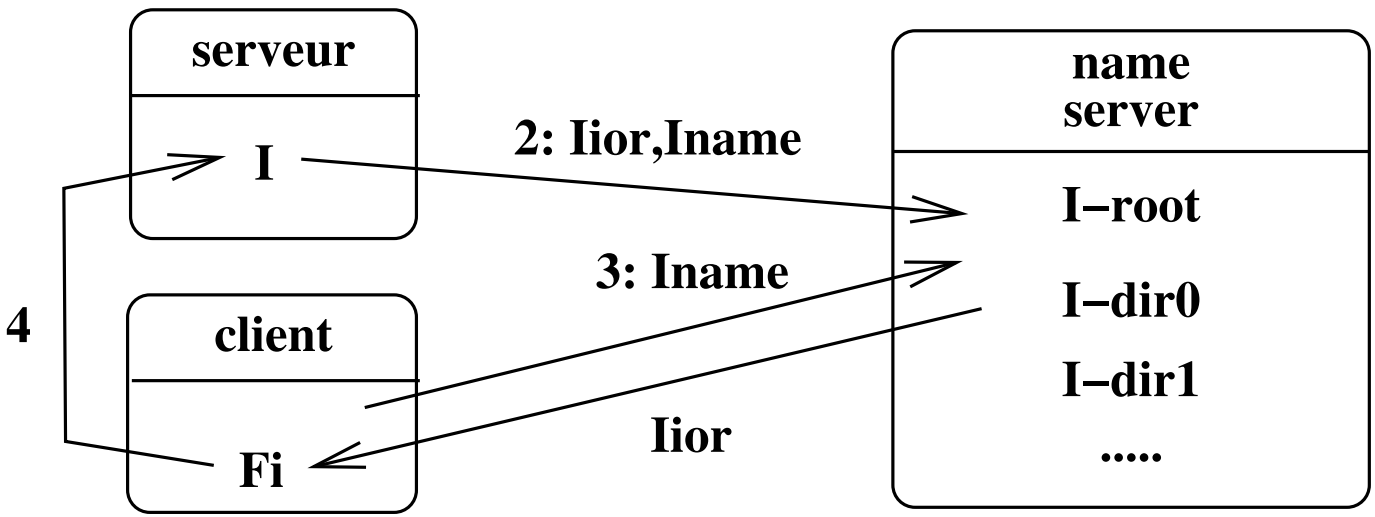
- structure arborescente (ex: $/rep_0/rep_1/.../nom$)
- nom de base: un couple (id.kind)

NS: Introduction

Principe

1. Un serveur gérant des couples (nom,IOR).
2. Les objets serveur s'enregistrent dans le NS sous un nom.
3. Les clients récupèrent du NS un objet en fonction du nom.

$\Longrightarrow$ NS est la racine du graphe d'objets.



Structure des noms

- structure arborescente (ex: $/rep_0/rep_1/.../nom$)
- nom de base: un couple (id.kind)

NS: Introduction

Principe

1. Un serveur gérant des couples (nom,IOR).
2. Les objets serveur s'enregistrent dans le NS sous un nom.
3. Les clients récupèrent du NS un objet en fonction du nom.

$\Longrightarrow$ NS est la racine du graphe d'objets.

Structure des noms

- structure arborescente (ex: $/rep_0/rep_1/.../nom$)
- nom de base: un couple (id.kind)

NS: IDL

```
module CosNaming {
  typedef string Istring;
  struct NameComponent {
    Istring id;
    Istring kind;
  };
  typedef sequence<NameComponent> Name;
  interface NamingContext {
    ...
  };
};
```

NS: IDL

```
module CosNaming {

  typedef string Istring;

  struct NameComponent {
    Istring id;
    Istring kind;
  };

  typedef sequence<NameComponent> Name;

  interface NamingContext {
    ...
  };

};
```

NS: IDL

```
module CosNaming {
```

```
  typedef string Istring;
  struct NameComponent {
    Istring id;
    Istring kind;
  };
```

```
  typedef sequence<NameComponent> Name;
```

```
  interface NamingContext {
    ...
  };
```

```
};
```

NS: IDL

```
module CosNaming {
```

```
  typedef string Istring;
  struct NameComponent {
    Istring id;
    Istring kind;
  };
```

```
  typedef sequence<NameComponent> Name;
```

```
  interface NamingContext {
    ...
  };
```

```
};
```

NS: IDL de NamingContext

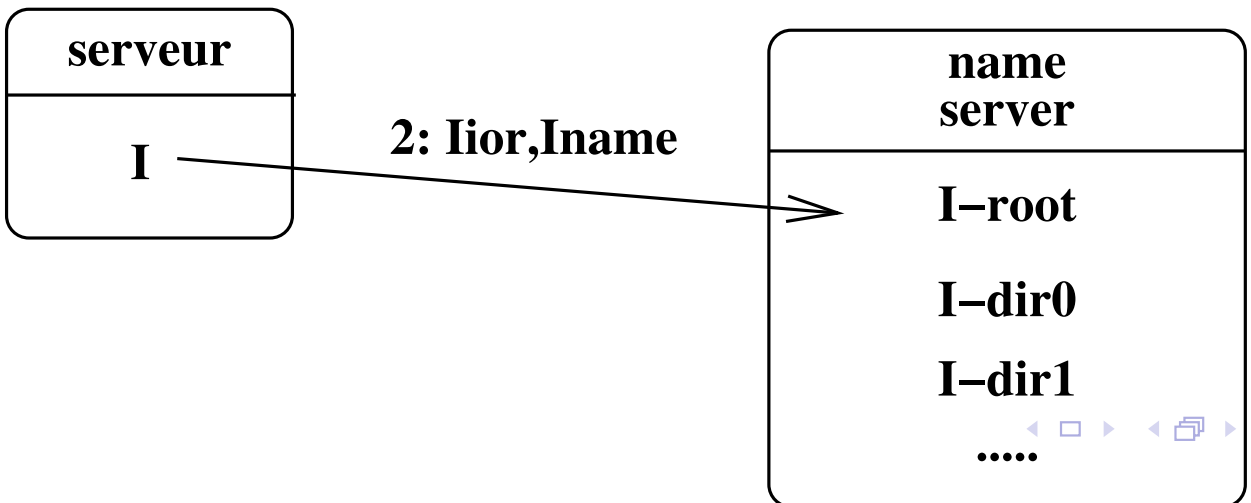
```
interface NamingContext {

};
```

NS: IDL de NamingContext

```
interface NamingContext {
```

```
  void bind(in Name n, in Object obj)
    raises(NotFound, CannotProceed, InvalidName, AlreadyBound);
   void rebind(in Name n, in Object obj)
    raises(NotFound, CannotProceed, InvalidName);
```

```
};
```

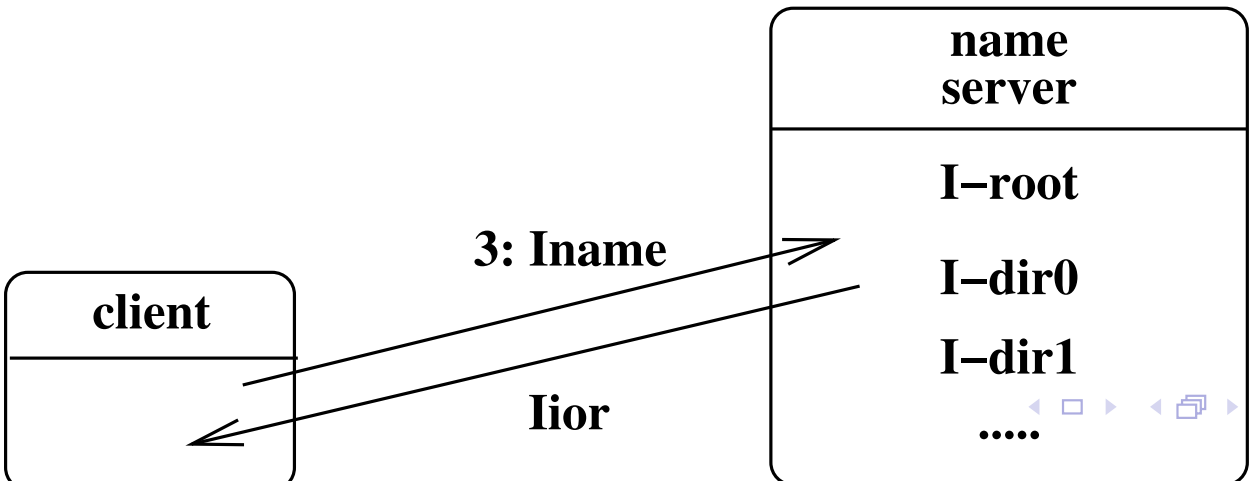


NS: IDL de NamingContext

```
interface NamingContext {
```

```
  Object resolve (in Name n)
    raises(NotFound, CannotProceed, InvalidName);
```

```
};
```



NS: IDL de NamingContext

```
interface NamingContext {
```

```
  void bind_context(in Name n, in NamingContext nc)
    raises(NotFound, CannotProceed, InvalidName, AlreadyBound)
   NamingContext bind_new_context(in Name n)
    raises(NotFound, AlreadyBound, CannotProceed, InvalidName)
```

```
};
```

NS: Utilisation

Lancement du serveur

Lancement d'une application

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > nameserv -i -OAport 5000    # ORBACUS
```

Lancement d'une application

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > tnameserv -ORBInitialPort 5000  # jdk
```

Lancement d'une application

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > tnameserv -ORBInitialPort 5000  # jdk
```

Lancement d'une application

```
unhost:bash > java app app-arg-0 app-arg-1 ...\
  --ORBInitialPort 5000 -ORBInitialHost lunix121
```

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > tnameserv -ORBInitialPort 5000  # jdk
```

Lancement d'une application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBservice  N-S iiop://lunix121:5000/DefaultNamingContext
```

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > tnameserv -ORBInitialPort 5000  # jdk
```

Lancement d'une application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > tnameserv -ORBInitialPort 5000  # jdk
```

Lancement d'une application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > tnameserv -ORBInitialPort 5000  # jdk
```

Lancement d'une application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > tnameserv -ORBInitialPort 5000  # jdk
```

Lancement d'une application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Utilisation

Lancement du serveur

```
lunix121:bash > tnameserv -ORBInitialPort 5000  # jdk
```

Lancement d'une application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

informations transmises

$\Longrightarrow$ le serveur est lancé sur un port donné.

$\Longrightarrow$ L'apllication récupère via N-S: host du server, port, le type de l'objet

NS: Programmation

Lancement de l'application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

Code de l'application

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  CosNaming::NamingContext_var sn = ...
  CosNaming::Name name = ...




}
```

Lancement de l'application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

Code de l'application

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  CosNaming::NamingContext_var sn = ...
  CosNaming::Name name = ...




}
```

Lancement de l'application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

Code de l'application

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  CosNaming::NamingContext_var sn = ...
  CosNaming::Name name = ...




}
```

```
  obj = orb -> resolve_initial_references("N-S");
  sn = CosNaming::NamingContext::_narrow(obj);
```

Lancement de l'application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

Code de l'application

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  CosNaming::NamingContext_var sn = ...
  CosNaming::Name name = ...




}
```

```
  name.length(1);
  name[0].id=   CORBA::string_dup("nom");
  name[0].kind= CORBA::string_dup("objet");
```

Lancement de l'application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

Code de l'application

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  CosNaming::NamingContext_var sn = ...
  CosNaming::Name name = ...
  // server
  MonObj_impl* o = new MonObj_impl();
  sn->rebind(name,o->_this());
}
```

Lancement de l'application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

Code de l'application

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  CosNaming::NamingContext_var sn = ...
  CosNaming::Name name = ...
  // client
  obj = sn->resolve(name);
  MonObj_var f = tx::MonObj::_narrow(obj);
}
```

PS: Fonction

Exemple

Une bibliothèque de livres, les attributs standards d'un livre sont: autheur, titre, genre, ...

Problème

Comment utiliser cette bibliothèque pour faire une bibliothèque spécifique:

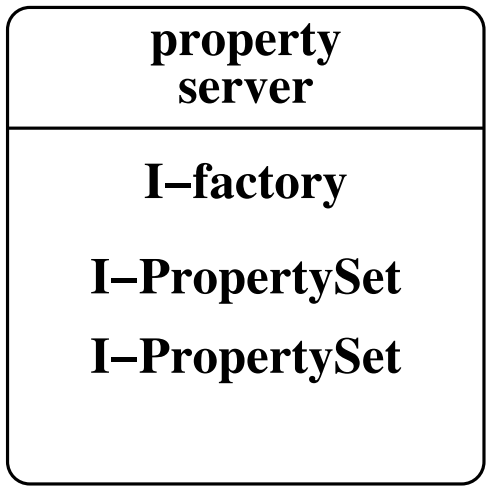
- personnelle en indiquand un jugement: coup de coeur, illisible, ...
- cormerciale en indiquant: prix, disponibilité, ...

Fonction

Associer dynamiquement des attributs aux objets.

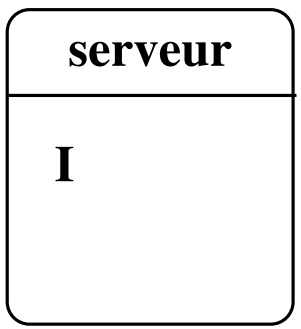
Principe

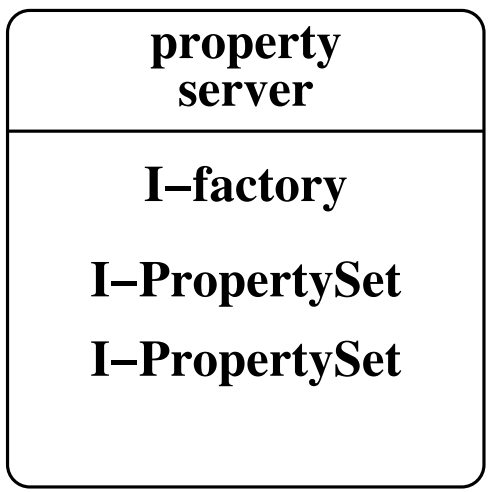
1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.



Principe

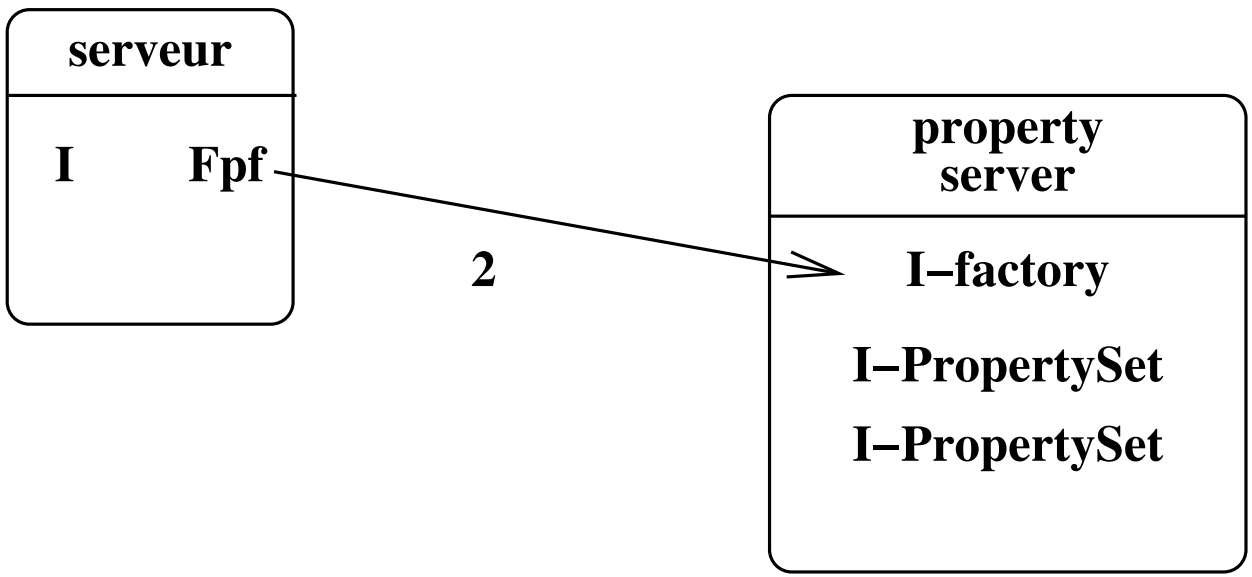
1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.





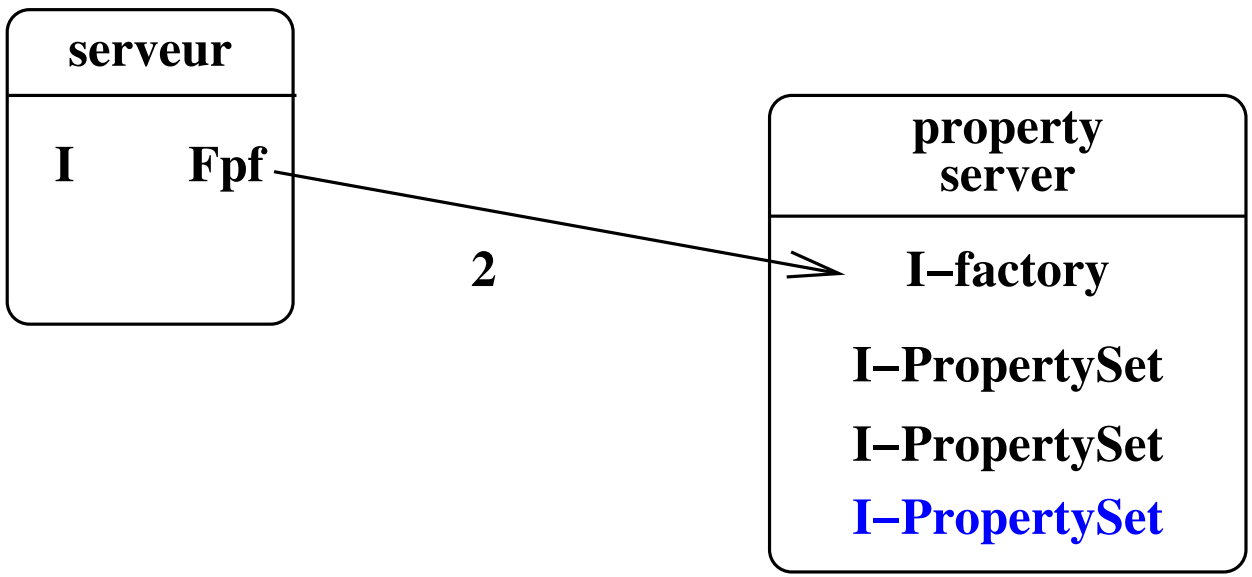
Principe

1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.



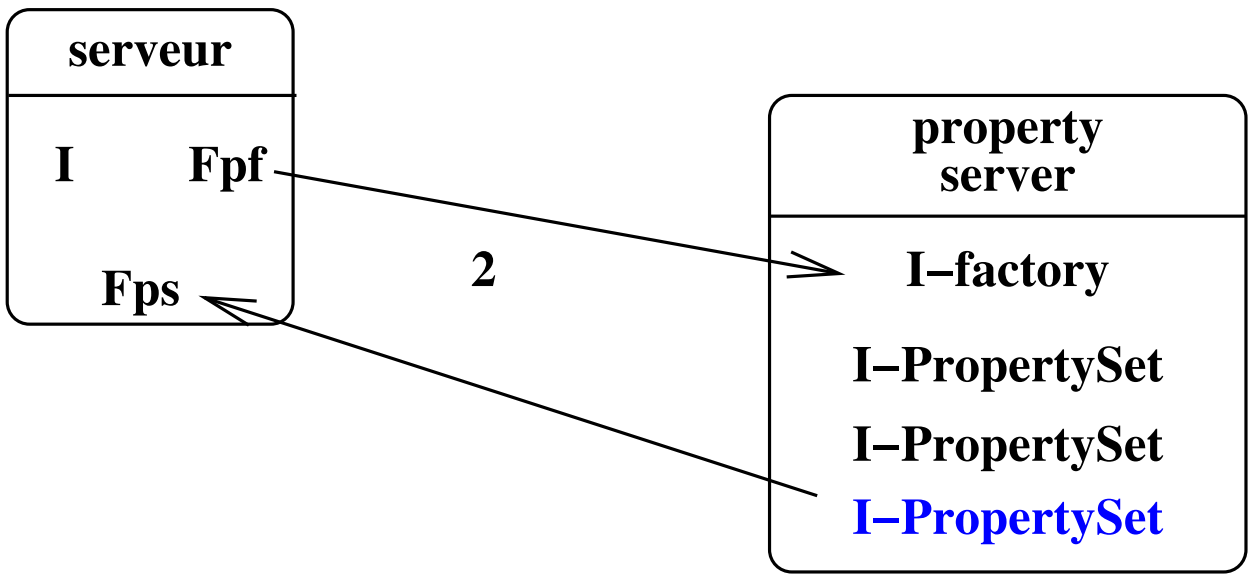
Principe

1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.



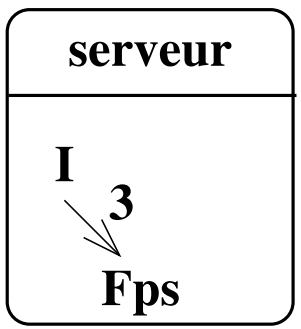
Principe

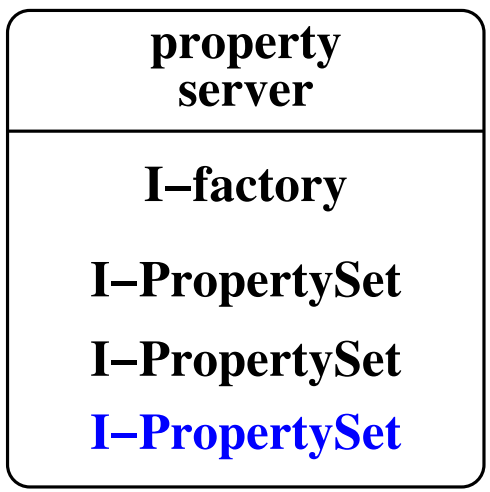
1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.



Principe

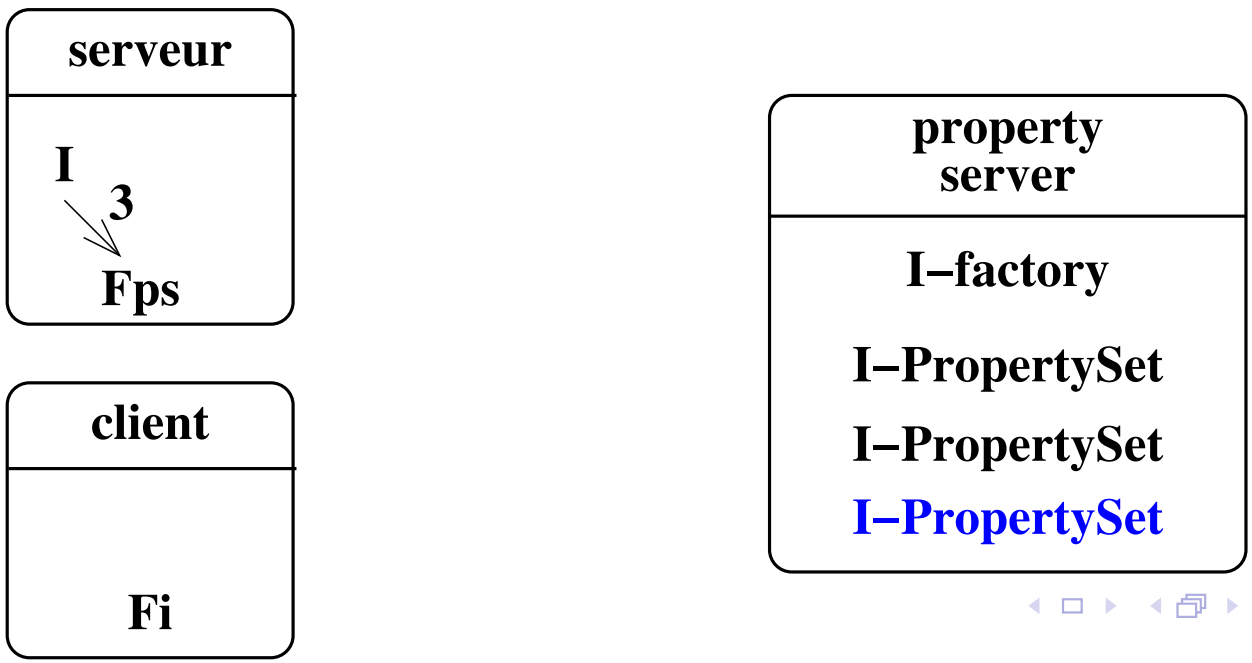
1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.





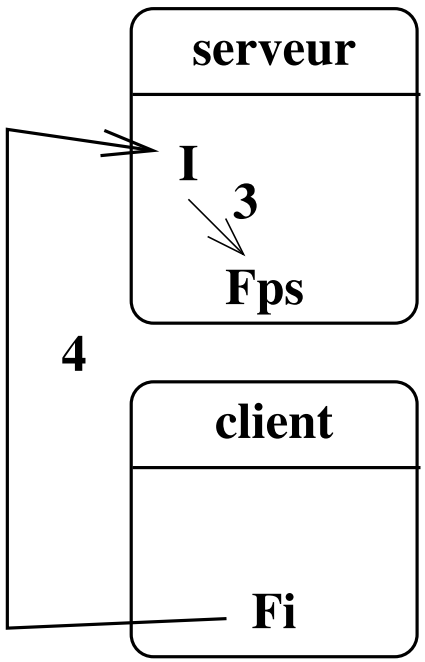
Principe

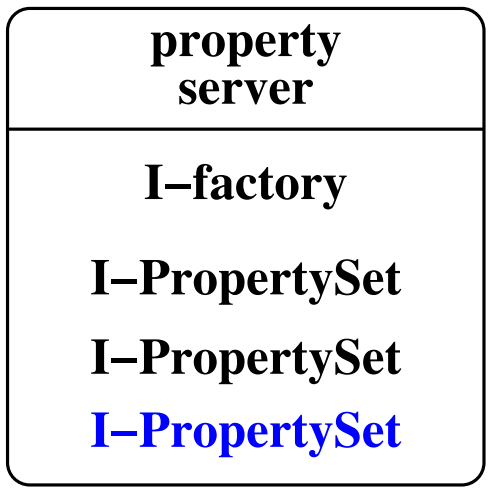
1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.



Principe

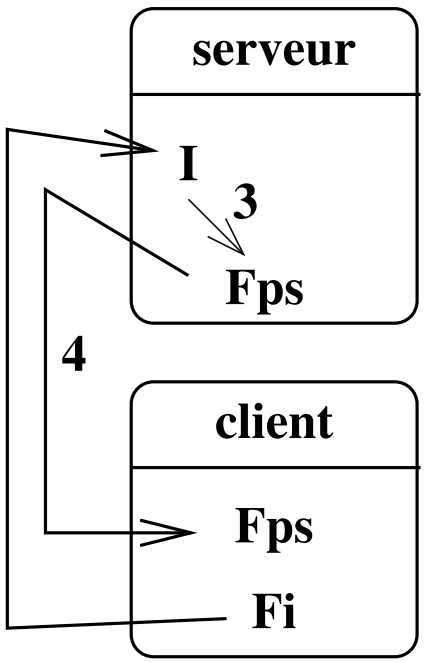
1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.

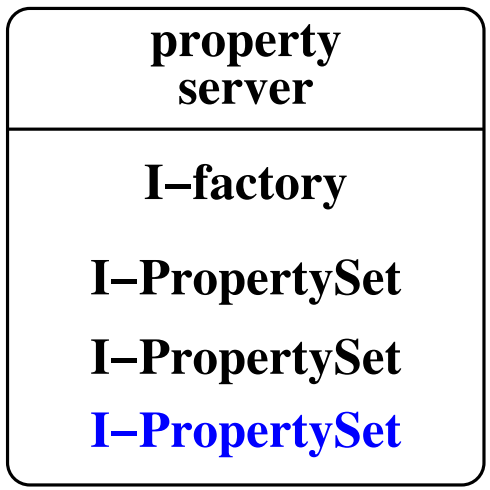




Principe

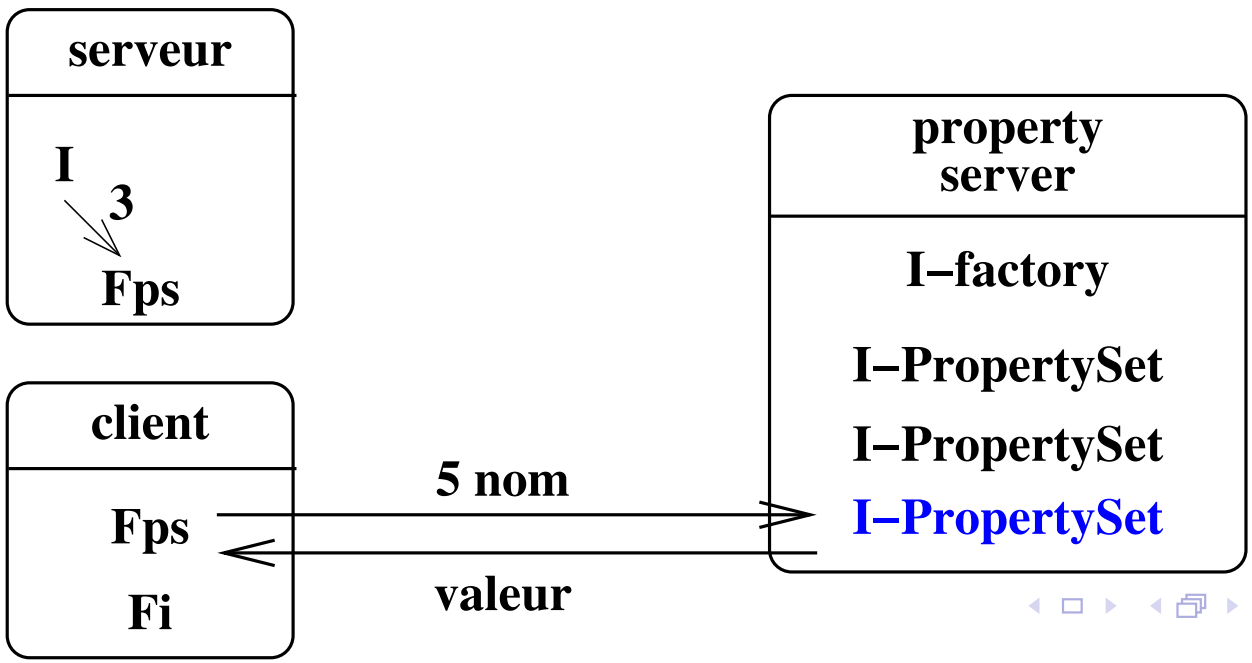
1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.





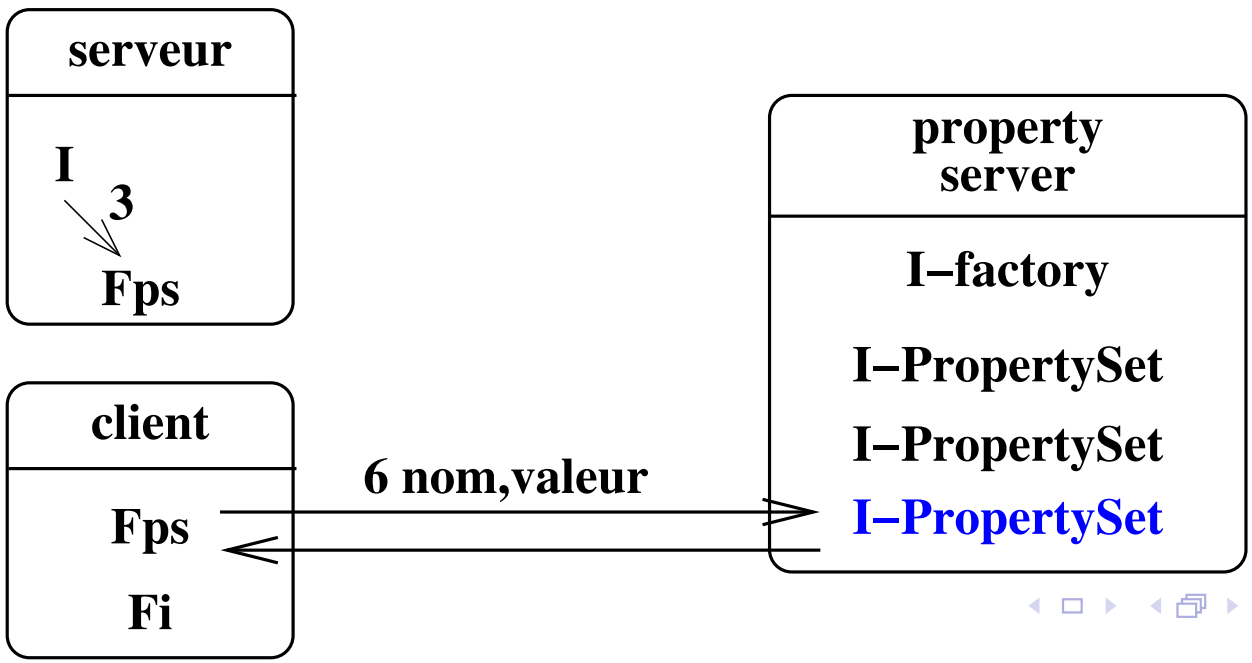
Principe

1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.



Principe

1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.



PS: Introduction

Principe

1. Un serveur gérant des ensemples de propiétés (couple: nom,valeur).
2. Un objet serveur demande un ensemble de proriétés.
3. Le serveur associe cette ensemble à son objet.
4. Un objet fantôme récupère l'ensemble de proriétés.
5. L'objet fantôme peut alors récupérer une propriété.
6. L'objet fantôme peut alors ajouter une propriété.

Structure des Propriétés

- proprité de base: un couple (nom.valeur)
- ensemble de couples $\{(n_0, v_0), (n_1, v_1), \ldots\}$.

PS: IDL

```
module PropertyService {

  typedef string PropertyName;
  struct Property {
    PropertyName property_name;
    any          property_value;
  };

  interface PropertySet        { ... };
  interface PropertySetFactory { ... };

  interface PropertySetDef : PropertySet { ... };
  interface PropertySetDefFactory { ... };

};
```

PS: IDL

```
module PropertyService {

  typedef string PropertyName;
  struct Property {
    PropertyName property_name;
    any          property_value;
  };

  interface PropertySet        { ... };
  interface PropertySetFactory { ... };

  interface PropertySetDef : PropertySet { ... };
  interface PropertySetDefFactory { ... };

};
```

PS: IDL

```
module PropertyService {
```

```
  typedef string PropertyName;
  struct Property {
    PropertyName property_name;
    any          property_value;
  };
```

```
  interface PropertySet        { ... };
  interface PropertySetFactory { ... };
```

```
  interface PropertySetDef : PropertySet { ... };
  interface PropertySetDefFactory { ... };
```

```
};
```

PS: IDL

```
module PropertyService {
```

```
  typedef string PropertyName;
  struct Property {
    PropertyName property_name;
    any          property_value;
  };
```

```
  interface PropertySet        { ... };
  interface PropertySetFactory { ... };
```

```
  interface PropertySetDef : PropertySet { ... };
  interface PropertySetDefFactory { ... };
```

```
};
```

PS: IDL de PropertySet

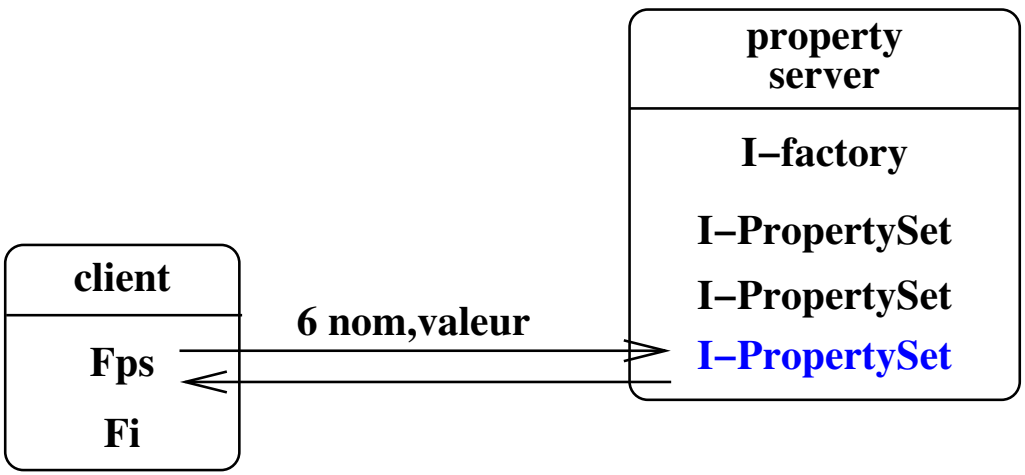
```
interface PropertySet {

};
```

PS: IDL de PropertySet

```
interface PropertySet {
```

```
  void define_property(
    in PropertyName property_name, in any property_value)
    raises(InvalidPropertyName, ConflictingProperty, ...);
```

```
};
```

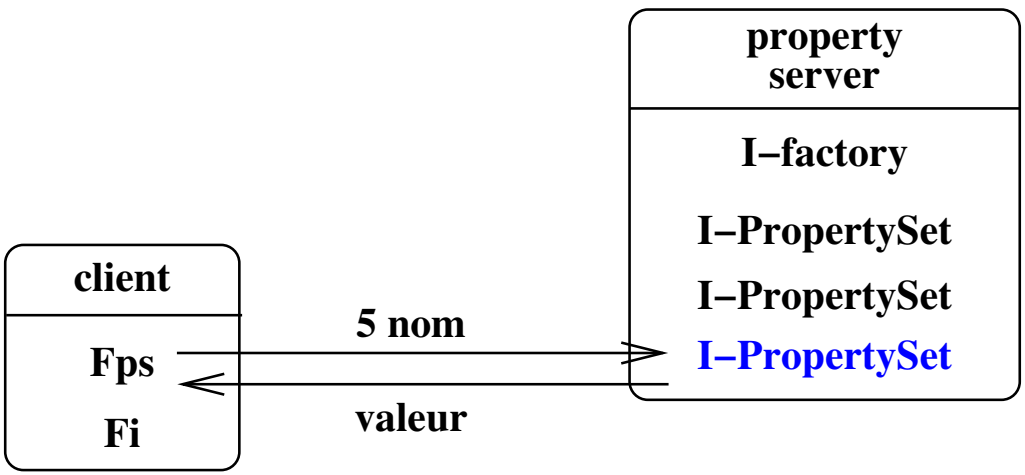


PS: IDL de PropertySet

```
interface PropertySet {
```

```
  boolean is_property_defined(in PropertyName property_name)
    raises(InvalidPropertyName);
  any  get_property_value(in  PropertyName property_name)
    raises(PropertyNotFound, InvalidPropertyName);
```

```
};
```



PS: IDL de PropertySet

```
interface PropertySet {
```

```
  void delete_property(in PropertyName property_name)
    raises(PropertyNotFound, InvalidPropertyName, ...);
  unsigned long get_number_of_properties();
  void get_all_property_names( ... )
```

```
};
```

PS: IDL de PropertySetFactory

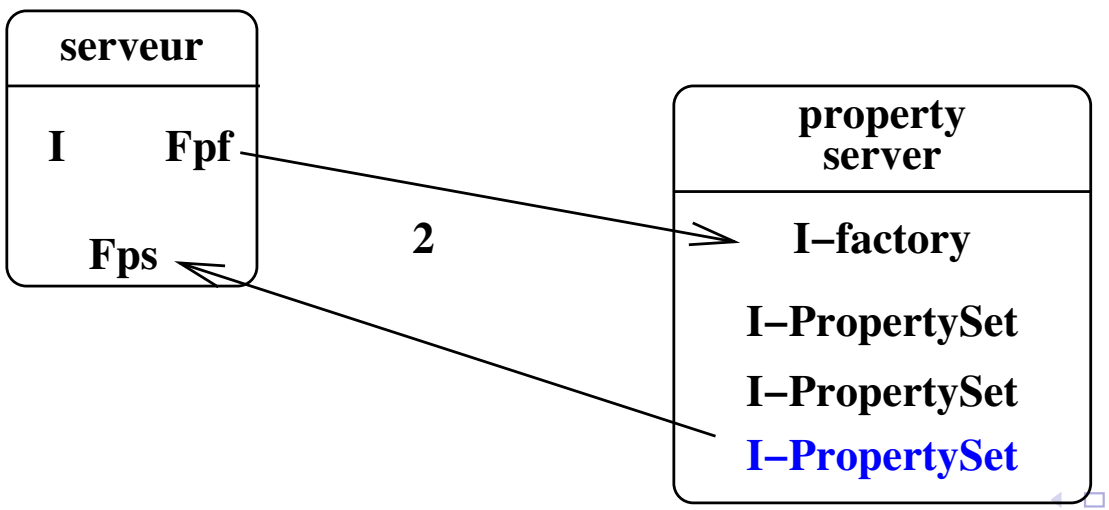
```
interface PropertySetFactory {

  PropertySet create_propertyset ();

  PropertySet create_initial_propertyset (
    in Properties initial_properties)
    raises (MultipleExceptions);

};
```

PS: IDL de PropertySetFactory

```
interface PropertySetFactory {
```

```
  PropertySet create_propertyset ();
  PropertySet create_initial_propertyset (
    in Properties initial_properties)
    raises (MultipleExceptions);
```

```
};
```



PS: Utilisation

Lancement du serveur

Lancement d'une application

PS: Utilisation

Lancement du serveur

```
lunix121:bash > propserv -i -OAport 6000   # ORBACUS
```

Lancement d'une application

PS: Utilisation

Lancement du serveur

```
lunix121:bash > propserv -i -OAport 6000   # ORBACUS
```

Lancement d'une application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBservice  P-S iiop://lunix121:6000/DefaultPropertySetFact
```

PS: Utilisation

Lancement du serveur

```
lunix121:bash > propertyd \
  --regname1 PropertySetFactory \
  --regname2 PropertySetDefFactory \
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

Lancement d'une application

PS: Utilisation

Lancement du serveur

```
lunix121:bash > propertyd \
  --regname1 PropertySetFactory \
  --regname2 PropertySetDefFactory \
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

Lancement d'une application

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBInitRef N-S=corbaloc::lunix121:5000/NameService
```

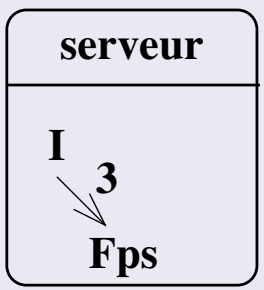
PS: Programmation

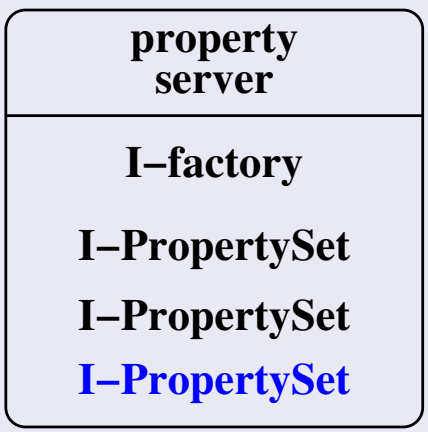
IDL

```
#include <PropertyService.idl>

interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

IDL

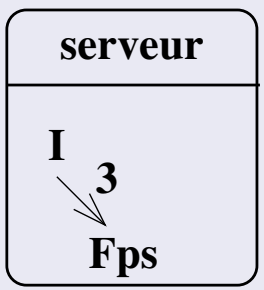
```
#include <PropertyService.idl>

interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

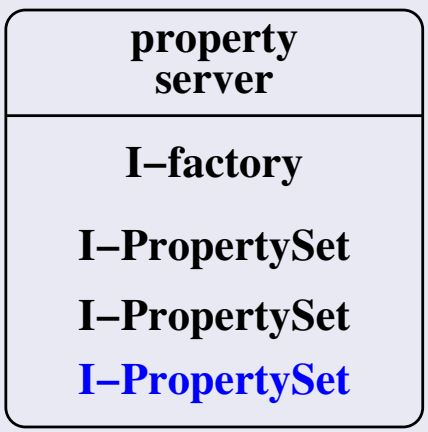




IDL

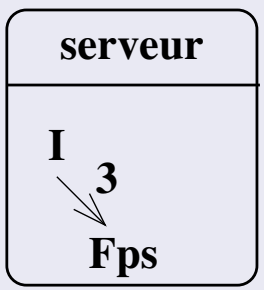
```
#include <PropertyService.idl>

interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

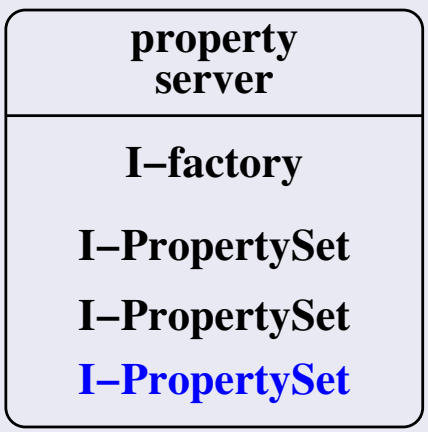




IDL

```
#include <PropertyService.idl>

interface book {
  string title();
  PropertyService::PropertySet properties();
};
```





PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

serveur

```
struct book_exp : public POA::book {
  PropertyService::PropertySet_var propset;
  char* title() { ... }
  PropertyService::PropertySet_ptr properties() {
    return propset->_duplicate(propset);
  }
  book_exp(PropertyService::PropertySetFactory_var psf) {
    propset = psf->create_propertyset();
  }
};
```

PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

serveur

```
struct book_exp : public POA::book {
  PropertyService::PropertySet_var propset;
  char* title() { ... }
  PropertyService::PropertySet_ptr properties() {
     return propset->_duplicate(propset);
  }
  book_exp(PropertyService::PropertySetFactory_var psf) {
    propset = psf->create_propertyset();
  }
};
```

PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

serveur

```
struct book_exp : public POA::book {
  PropertyService::PropertySet_var propset;
  char* title() { ... }
  PropertyService::PropertySet_ptr properties() {
     return propset->_duplicate(propset);
  }
  book_exp(PropertyService::PropertySetFactory_var psf) {
    propset = psf->create_propertyset();
  }
};
```

PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

serveur

```
struct book_exp : public POA::book {
  PropertyService::PropertySet_var propset;
  char* title() { ... }
  PropertyService::PropertySet_ptr properties() {
     return propset->_duplicate(propset);
  }
  book_exp(PropertyService::PropertySetFactory_var psf) {
    propset = psf->create_propertyset();
  }
};
```

PS: Programmation

server

```
struct book_exp : public POA::book {
  ...
  book_exp(PropertyService::PropertySetFactory_var psf);
};

int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa = ...;
  PropertyService::PropertySetFactory_var psf = ...
  book_exp* obj = new book(psf);
  ...
  poa->the_POAManager()->activate();
  orb->run();
}
```

PS: Programmation

server

```
struct book_exp : public POA::book {
  ...
  book_exp(PropertyService::PropertySetFactory_var psf);
};

int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa = ...;
  PropertyService::PropertySetFactory_var psf = ...
  book_exp* obj = new book(psf);
  ...
  poa->the_POAManager()->activate();
  orb->run();
}
```

PS: Programmation

server

```
struct book_exp : public POA::book {
  ...
  book_exp(PropertyService::PropertySetFactory_var psf);
};

int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa = ...;
  PropertyService::PropertySetFactory_var psf = ...
  book_exp* obj = new book(psf);
  ...
  poa->the_POAManager()->activate();
  orb->run();
}
```

PS: Programmation

server

```
struct book_exp : public POA::book {
  ...
  book_exp(PropertyService::PropertySetFactory_var psf);
};

int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa = ...;
  PropertyService::PropertySetFactory_var psf = ...
  book_exp* obj = new book(psf);
  ...
  poa->the_POAManager()->activate();
  orb->run();
}
```

PS: Programmation

server

```
struct book_exp : public POA::book {
  ...
  book_exp(PropertyService::PropertySetFactory_var psf);
};

int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa = ...;
  PropertyService::PropertySetFactory_var psf = ...
  book_exp* obj = new book(psf);
  ...
  poa->the_POAManager()->activate();
  orb->run();
}
```

PS: Programmation

server

```
struct book_exp : public POA::book {
  ...
  book_exp(PropertyService::PropertySetFactory_var psf);
};

int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa = ...;
  PropertyService::PropertySetFactory_var psf = ...
  book_exp* obj = new book(psf);
  ...
  poa->the_POAManager()->activate();
  orb->run();
}
```

```
unhost:bash > app app-arg-0 app-arg-1 ...\
  -ORBservice P-S iiop://lunix121:6000/DefaultPropertySetFacto
```

PS: Programmation

server

```
struct book_exp : public POA::book {
  ...
  book_exp(PropertyService::PropertySetFactory_var psf);
};

int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  PortableServer::POA_var poa = ...;
  PropertyService::PropertySetFactory_var psf = ...
  book_exp* obj = new book(psf);
  ...
  poa->the_POAManager()->activate();
  orb->run();
}
```

```
  obj = orb -> resolve_initial_references("P-S");
  psf = PropertyService::PropertySetFactory::_narrow(obj);
```

PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // utilisation des proprités
}
```

PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // utilisation des proprités
}
```

PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // utilisation des proprités
}
```

PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // utilisation des proprités
}
```

PS: Programmation

IDL

```
interface book {
  string title();
  PropertyService::PropertySet properties();
};
```

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // utilisation des proprités
}
```

PS: Programmation

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // utilisation des proprités
}
```

PS: Programmation

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // utilisation des proprités
}
```

```
  long        note;
  CORBA::Any any ;
```

PS: Programmation

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // écriture d'une proprité
}
```

```
  long        note;
  CORBA::Any any ;
  note=12; any <<= note;
   ps->define_property("note-ia",any);
```

PS: Programmation

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // écriture d'une proprité
}
```

```
  long        note;
  CORBA::Any any ;
  note=12; any <<= note;
   ps->define_property("note-ia",any);
```

PS: Programmation

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // obtention d'une proprité
}
```

```
  long        note;
  CORBA::Any any ;
  any=*ps->get_property_value("note-ia");
  any >>= note;
```

PS: Programmation

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // obtention d'une proprité
}
```

```
  long        note;
  CORBA::Any any ;
  any=*ps->get_property_value("note-ia");
  any >>= note;
```

PS: Programmation

client

```
int main(int argc, char** argv) {
  CORBA::ORB_var orb= CORBA::ORB_init(argc,argv);
  book_var* book = ...;
  PropertyService::PropertyService_var ps=book->propertie();
  // utilisation des proprités
}
```

- <<= surchargé pour les principaux types
  `any <<= (char*) ; any <<= (double) ;`
  `any <<= (CORBA::Object*) ; ...`
- >>= surchargé pour les principaux types
  `any >>= (const char*) ; any >>= (double) ;`
  `any >>= (CORBA::Object*) ; ...`

Table des matières

Vrai middleware

Définition d'un middleware

Faire comuniquer des processus en:

- abstraction de la location physique
- abstraction des supports de communication
- abstraction des PDUs
- abstraction des langages

CORBA

Une fois que l'on a une racine du graphe d'objet,
$\Longrightarrow$ on passe d'un objet à l'autre de facon 100% transparente.

Vrai middleware

Définition d'un middleware

Faire comuniquer des processus en:

- abstraction de la location physique
- abstraction des supports de communication
- abstraction des PDUs
- abstraction des langages

CORBA

Une fois que l'on a une racine du graphe d'objet,

$\Longrightarrow$ on passe d'un objet à l'autre de facon 100% transparente.

Qu'est ce CORBA?

Specification fonctionnelle

- une description du moteur.
- la définition de l'IDL et du format des IORs.
- une description du mapping de l'IDL vers les langages courants.
- un protocole pour permettre l'interopérabilité des ORBs IIOP (Internet Inter-ORB Protocol).
- une boite à outils généraux (services).
  Ce sont des descriptions IDL, documentées
- des boites à outils métiers Ce sont des descriptions IDL, documentées

ORB CORBA compliant

Une implantation (propriétaire) respectant les spécifications.
Elles sont appelées ORB (ex: ORB Mico, ORB Orbix, ...).

Propriétés

Interopérabilité d'ORB

$\Longrightarrow$ une aplication CORBA tourne sur n'importe implantation CORBA.

$\Longrightarrow$ 2 aplications CORBA développées sur le même ORB peuvent comuniquer.

$\Longrightarrow$ 2 applications CORBA développées sur des ORBs différents peuvent communiquer.

Protocole ouvert

D'autres protocoles peuvent être définis en plus de l'IIOP. Ceci permet de faire un ORB:

- optimisé pour mono machine.
- optimisé pour un ensemble de machine identiques.
- optimisé pour un protocole de transport.

Domaines d'applications

Domaines

- $\Longrightarrow$ Milieu hétérogène.
- $\Longrightarrow$ Existence d'un ORB pour les langages cibles.
- $\Longrightarrow$ Pas de contraintes de performance réseau.

Applications

- $\Longrightarrow$ Gestionnaire de fermes de calcul.
- $\Longrightarrow$ Robotique/Controleur
- $\Longrightarrow$ web & applet
- $\Longrightarrow$ window manager
- $\Longrightarrow$ widget manager

Domaines d'applications

Domaines

- $\Longrightarrow$ Milieu hétérogène.
- $\Longrightarrow$ Existence d'un ORB pour les langages cibles.
- $\Longrightarrow$ Pas de contraintes de performance réseau.

Applications

- $\Longrightarrow$ Gestionnaire de fermes de calcul.
- $\Longrightarrow$ Robotique/Controleur
- $\Longrightarrow$ web & applet
- $\Longrightarrow$ window manager
- $\Longrightarrow$ widget manager