
RPC

Remote procedure call

AB – v1.29 (13/10/2021)

Présentation des RPC

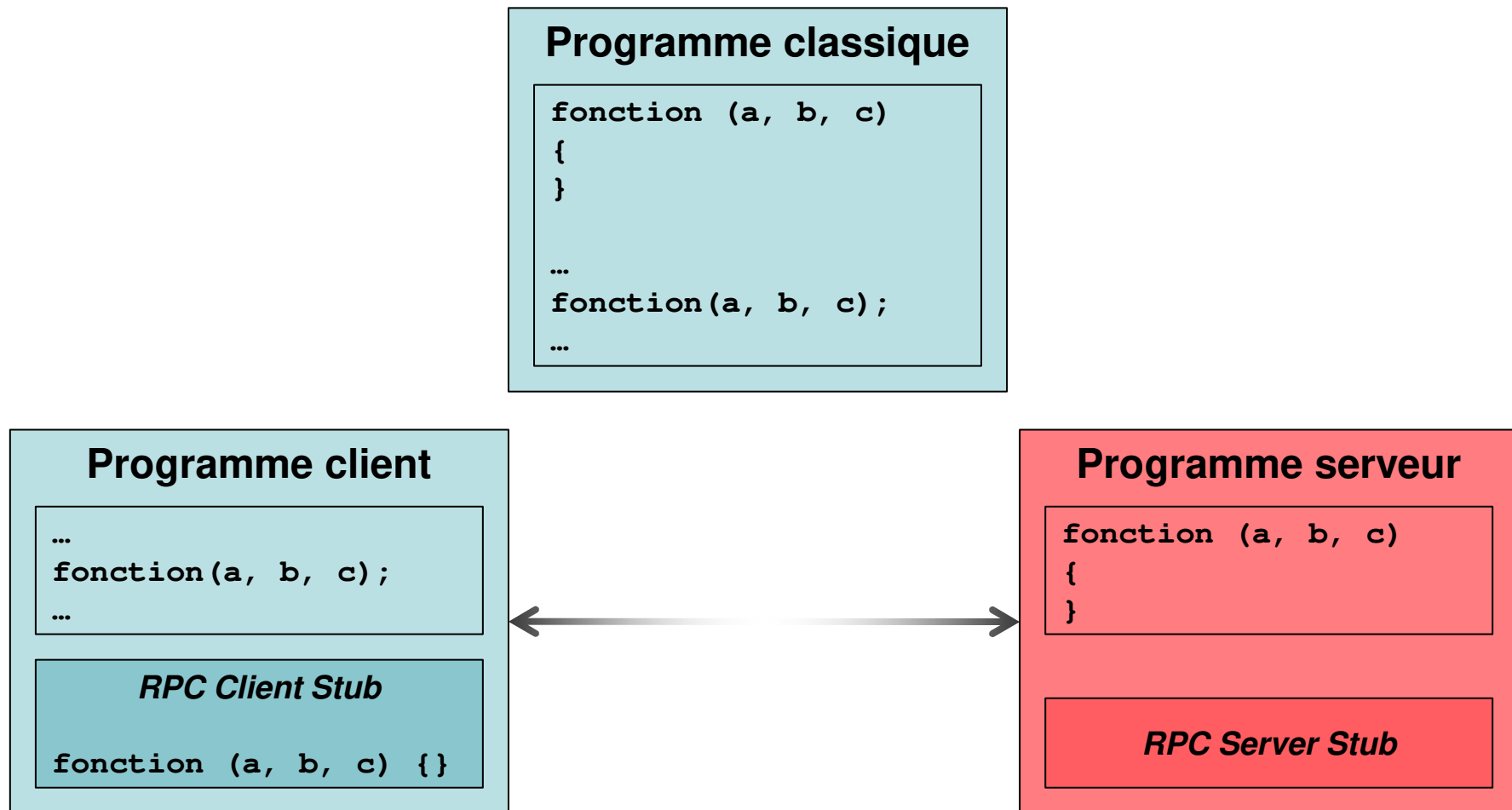
Remote Procedure Call

- Mécanisme d'IPC (*Interprocess Communication*), inclus dans Windows, permettant la mise en œuvre de programmes Client/Serveur
- De nombreuses fonctionnalités (communications réseau, authentification, ...) sont gérées par le *run-time* RPC (bibliothèque rpcrt4.dll) et sont totalement transparentes pour les applications
- Développement très lié au langage C/C++

Principe de base

- RPC permet l'appel, dans un processus client, d'une fonction s'exécutant dans un autre processus (dit serveur)

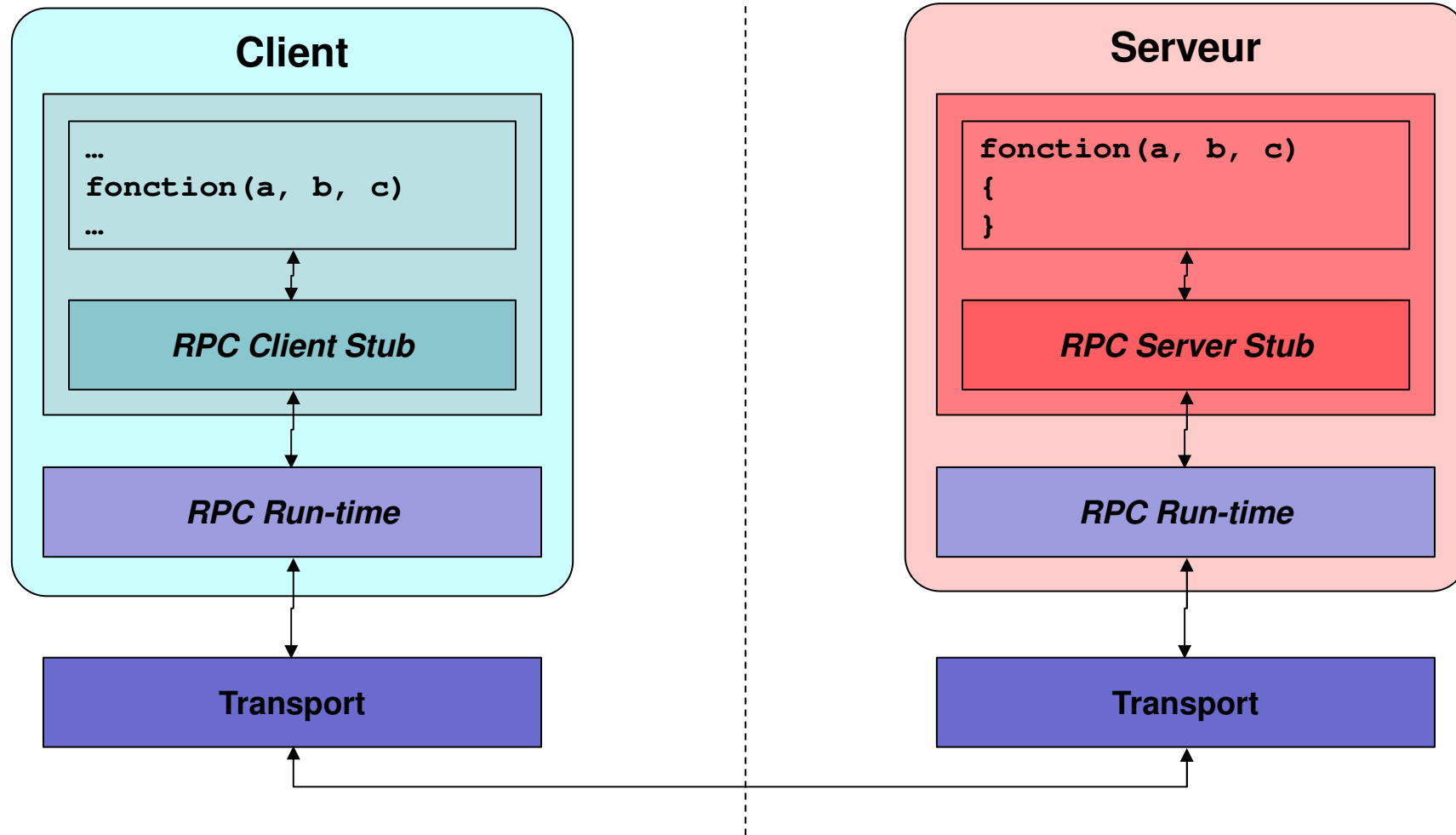
Schéma d'appel



Client Stub

- Lorsque le client appelle une fonction, il appelle en réalité une fonction du *stub* client
- Le rôle du *stub* client est de :
 - Valider les paramètres passés à la fonction
 - Convertir ces paramètres au format NDR (*Network Data Representation*)
 - Transmettre au *Run-time* RPC l'appel avec les paramètres

Schéma d'appels



Interfaces et IDL

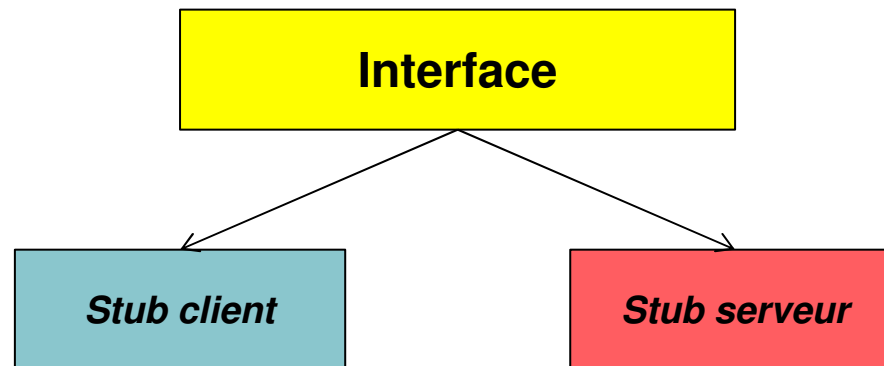
Interfaces

- Les fonctions RPC sont regroupées au sein d'une **interface**
- Une interface est caractérisée par :
 - un identifiant unique (UUID)
 - un numéro de version
- Une interface définit :
 - les variables exportées : nom et type
 - les fonctions exportées : nom, paramètres et valeur de retour

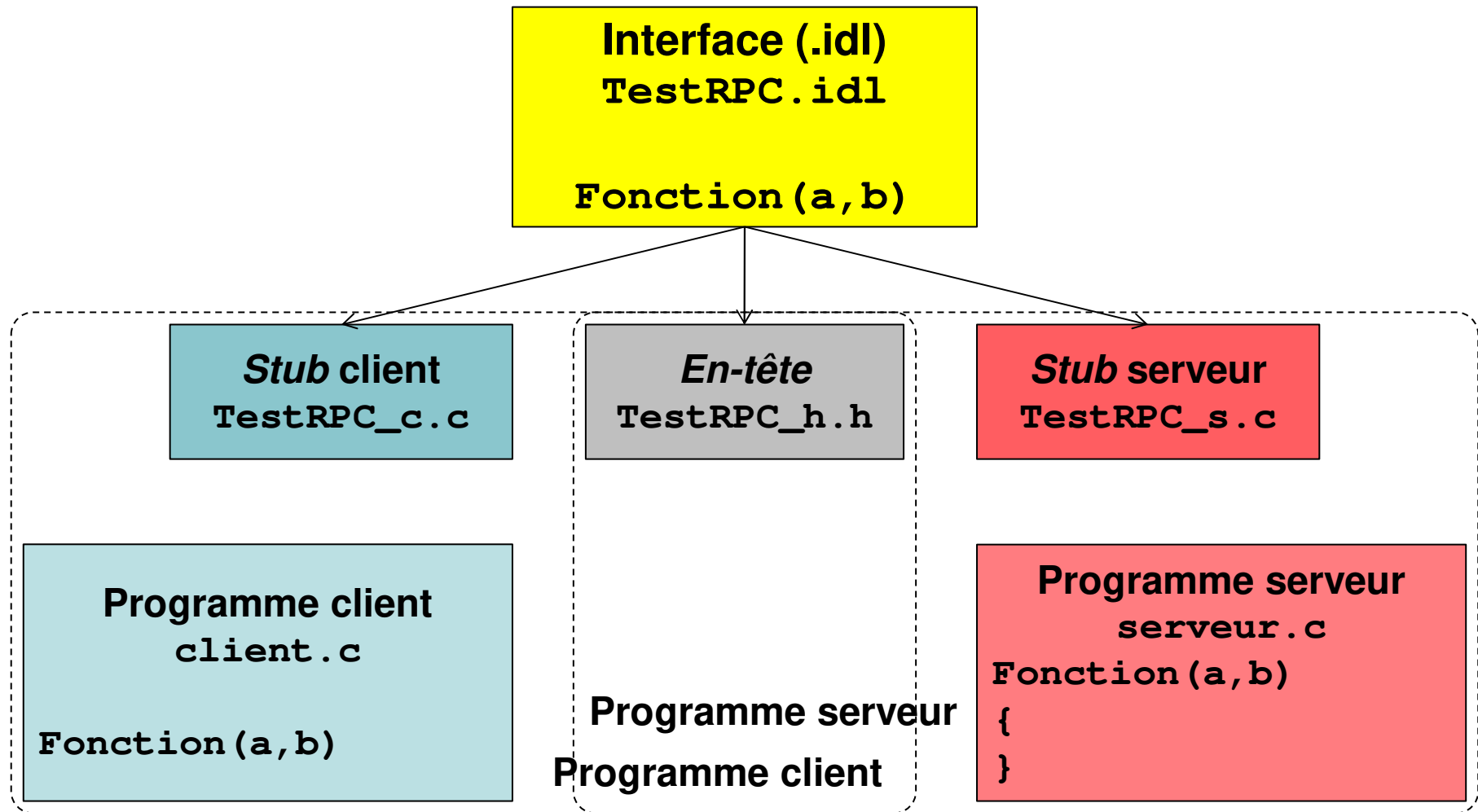
MIDL

Microsoft Interface Definition Language

- Langage de définition des interfaces RPC
- La compilation d'un fichier IDL génère :
 - Un fichier d'en-tête commun
 - Le stub client
 - Le stub serveur



Génération des programmes



Structure d'un fichier IDL

- En tête de déclaration de l'interface :

```
[  
    uuid(DF0BA905-0B89-41E4-99A8-3AC166C8C3D1),  
    version(1.0)  
]
```

- Définition des fonctions de l'interface:

```
interface NomInterface  
{  
    Fonction1();  
    Fonction2();  
    ...  
}
```

Typage des données

- IDL permet de typer fortement les variables :
 - Types de base : **bool**, **byte**, **int**, **long**, **char**, **wchar_t**, ...
 - Structure : **struct**
 - Union : **typedef union**
 - Énumération : **enum**
 - Tableau

Attributs

- Il est également possible de spécifier des attributs sur l'interface, les fonctions ou les paramètres :
 - `[in] [out] [in, out]`
 - `[ref]`
 - `[string]`
 - `[min_is(arg)] [max_is(arg)]`
 - `[range(0, arg)]`
 - `[size_is(arg)] [length_is(arg)]`

Protocoles de transport

Binding handles

Binding Handle

RPC_BINDING_HANDLE / handle_t

- Le *Binding* représente une connexion entre un client et un serveur
 - La connexion et son état associé sont représentés par un *Binding Handle* (handle de connexion)
 - 3 types possibles :
 - ***implicit*** : le *binding* est créé et utilisé implicitement à chaque appel de fonction RPC
 - ***explicit*** : le *binding* est créé et doit être spécifié en argument à chaque appel de fonction RPC
 - ***automatic*** : le *binding* est entièrement géré par le *run-time* RPC
-

Principaux protocoles réseau

Nom	Type
ncalrpc	Network Computing Architecture local remote procedure call
ncacn_x	Network Computing Architecture connection-oriented protocol
ncadg_x	Network Computing Architecture datagram protocol

Nom	Type	Option
ncalrpc	<i>Local Inter-Process Communication port</i>	Nom du LPC
ncacn_ip_tcp	Socket TCP	Port TCP
ncacn_np	Canaux nommés	Nom du canal nommé
ncacn_http	HTTP	Port et proxy
ncadg_ip_udp	Datagramme UDP	Port UDP

Mise en œuvre

Mise en œuvre côté serveur

- Déclaration du protocole réseau à utiliser :
 - **RpcServerUseProtseq**
- Enregistrement de l'interface RPC :
 - **RpcServerRegisterIf**
- Mise en écoute de l'interface :
 - **RpcServerListen**

Mise en œuvre côté client

- Création d'un *Binding Handle*
 - **RpcStringBindingCompose**
 - **RpcBindingFromStringBinding**
- Appel des fonctions RPC avec la mise en place d'un gestionnaire d'exception :
 - **RpcTryExcept**
 - **RpcExcept**
 - **RpcEndExcept**

Authentication

Authentication

- Les authentications sont réalisées via les SSP
- Il est possible (ou nécessaire) de définir (ou d'imposer) :
 - un **service d'authentification**
 - **RPC_C_AUTHN_X**
 - un **niveau d'authentification**
 - **RPC_C_AUTHN_LEVEL_X**
 - un **service d'autorisation**
 - **RPC_C_AUTHZ_X**

Constantes d'authentification

RPC_C_AUTHN_X	RPC_C_AUTHN_LEVEL_X	RPC_C_AUTHZ_X
RPC_C_AUTHN_NONE	RPC_C_AUTHN_LEVEL_NONE	RPC_C_AUTHZ_NONE
RPC_C_AUTHN_WINNT	RPC_C_AUTHN_LEVEL_CONNECT	
RPC_C_AUTHN_GSS_KERBEROS	RPC_C_AUTHN_LEVEL_PKT_INTEGRITY	
RPC_C_AUTHN_GSS_NEGOTIATE	RPC_C_AUTHN_LEVEL_PKT_PRIVACY	
RPC_C_AUTHN_DEFAULT	RPC_C_AUTHN_LEVEL_DEFAULT	RPC_C_AUTHZ_DEFAULT

Authentication côté serveur

- Le serveur peut demander, **mais sans l'exiger**, l'authentification du client (**RpcServerRegisterAuthInfo**) :
 - Les clients fournissant une authentification valide seront **acceptés**
 - Les clients fournissant une authentification invalide seront **rejetés**
 - Les clients ne fournissant pas d'authentification seront **acceptés**
- C'est à la fonction RPC de vérifier si l'authentification a bien eu lieu

Authentication côté client

- Le client peut demander à s'authentifier auprès du serveur (**RpcBindingSetAuthInfo**)
- L'authentification cliente peut-être :
 - **Implicite** : utilisation du contexte de la session d'authentification
 - **Explicite** : nécessite de fournir des *credentials*

Bibliothèque de protections RPC

- Le paramètre **AuthnSvc** spécifie l'identifiant du *package* d'authentification réseau à utiliser
- Chaque SSP dispose d'un identifiant RPC :
`SecPkgInfo.wRPCID` `// ID for RPC Runtime`
- La configuration RPC permet de spécifier des SSP spécifiques à RPC :
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Rpc\SecurityService`

Mécanismes de sécurité

Security Callback

- Côté serveur, lors de l'enregistrement d'une interface, il est possible de définir une fonction de rappel de sécurité (*security callback*)
- Cette fonction sera invoquée avant tout appel à une fonction RPC d'une interface
- Le *callback* permet généralement de :
 - Valider l'authentification du client
 - Valider le protocole réseau utilisé par le client

Protection contre les accès anonymes

- Seuls les canaux **ncacn_np** et **ncalrpc** peuvent être protégés via un descripteur de sécurité
- Pour assurer l'authentification, il faut soit :
 - Utiliser un *callback*, ce qui impose l'authentification du client sauf si l'option **RPC_IF_ALLOW_CALLBACKS_WITH_NO_AUTH** est positionnée
 - Positionner **RPC_IF_ALLOW_SECURE_ONLY**