

PA160

**Distribuované aplikace – základní protokoly
(Programování distribuovaných systémů)**

RPC

- RPC – Remote Procedure Call
 - Tváří se jako „obyčejná“ procedura
 - Tělo je vykonáno v jiném procesu (na jiném stroji]
- Důvody zavedení
 - Vyšší úroveň abstrakce než sokety
 - Nezávislé na architektuře či operačním systému
 - Předávání jednoduchých i složitých datových typů
- Vyžaduje podporu (jmenné služby, bezpečnost, ...)

RPC Standardy

- Tři základní
 - ONC (Open Network Computing)
 - DCE (Distributed Computing Environment)
 - Microsoftí COM/DCOM „standard“

RPC Standardy – popis

- **ONC** (definováno v RFC 1831)
 - vyvinuto SUNem
 - nejrozšířenější, dostupné prakticky na všech operačních systémech
- **DCE** (<http://www.opengroup.org/dce/>)
 - Rovněž široce dostupné
 - Zahrnuje komplexní služby
 - Postupně rozšířeno o objektový přístup
- **COM** (<http://www.microsoft.com/com/>)
 - Proprietární
 - V současné době rozšíření DCE, původně componentní model (jeden počítač)

ONC RPC

- RPC jedinečně definován:
 - *Číslem programu*: skupina procedur
 - *Číslem verze*: verze
 - *Číslem procedury*: jedinečný identifikátor konkrétní procedury
 - Uživatel může přiřazovat čísla programů v rozsahu 0x20000000--0x3FFFFFFF (20 milionů čísel)

ONC – části

■ Clientská část (příklad)

- `clnt_create()`: de facto napojení na skupinu procedur (server)
- `clnt_call()`: zavolání konkrétní procedury
- `clnt_destroy()`: odpojení se

■ Serverová část (příklad)

- `svc_register()`: registruj skupinu u portmapperu
- `svc_getargs()`: převezmi argumenty (XDR zakódované)
- `svc_sendreply()`: zašli výsledek volání (XDR)
- `svc_unregister()`: zruš registraci

Portmapper

- Zprostředkuje přístup ke službám
 - Naslouchá na portu 111
 - Udržuje mapu programů a odpovídajících čísel portů
 - Použití
 - * 1. Server se zaregistruje na portmapperu
 - * 2. Client získá aktuální číslo portu konkrétní služby od portmapperu (dotaz na portu 111)
 - * 3. Client se spojí se serverem

ONC API – tvorba klienta

CLIENT *

```
clnt_create(host, prog, vers, prot)
```

```
    char *host;      -- hostname
```

```
    u_long prog;     -- program number
```

```
    u_ong vers;      -- version number
```

```
    char *prot;      -- protocol (udp, tcp, unix)
```

- Vrací *handle* na skupinu procedur

ONC API – registrace serveru

```
extern bool_t
svc_register(xprt, prog, vers, dispatch, protocol)
    SVCXPRT *xprt;
    u_long prog;
    u_long vers;
    void (*dispatch)();
    u_long protocol; //tcp or udp, zero means do not register
```

- Registruje *program* a jeho *verzi*
- Vrací true při úspěchu

ONC API – volání klienta

```
enum clnt_stat
CLNT_CALL(rh, proc, xargs, argsp, xres, resp, timeout)
    CLIENT *rh; -- handle from clnt_create
    u_long proc; -- registered procedure
    xdrproc_t xargs; -- XDR to encode input
    caddr_t argsp; -- address of input agr
    xdrproc_t xres; -- XDR to decode output
    caddr_t resp; -- address of result buffer
    struct timeval timeout; -- timeout
```

XDR

- Problém s odlišnou architekturou komunikujících počítačů
 - Pořadí bytů, typy čísel, řetězce, enumerované typy
- XDR: eXchange Data Representation funkce
 - Kódují a dekódují data
 - Konkrétní implementace je závislá na architektuře
 - Zajišťují strojovou nezávislost při výměně dat

Vestavěné XDR konverzní funkce

Používány v `clnt_call()`
Jednoduché funkce (příklady)

```
xdr_int()    xdr_char()  xdr_u_short   xdr_bool()  
xdr_long()  xdr_u_int() xdr_wrapstring()  
xdr_short() xdr_enum()  xdr_void()
```

K dispozici jsou i agregační funkce (příklady)

```
xdr_array()  xdr_string()  xdr_union()  
xdr_vector() xdr_opaque()
```

Rovněž je možno definovat uživatelské funkce umožňující manipulovat s nepředdefinovanými typy

Generování kódu pro použití RPC

- Nevýhody nativního RPC

- Složitá API, těžko laditelné
- Použití XDR složité

Většinou se používá stále stejným způsobem

- Řešení: použití generátoru kódu (`rpcgen`).

Generuje:

- Hlavičkový soubor
- Potřebné XDR funkce
- *stubs* na straně klienta i serveru

- `rpcgen` pro C/C++, `jrpcgen` pro Javu

Chování RPC

- Neúspěchy
 - Nelze nalézt server
 - Ztracený požadavek
 - Ztracený výsledek
 - Zhroucení serveru (po přijetí požadavku)
 - Zhroucení klienta

Ztracený požadavek

- Používání timeoutů
- Používání retransmise
 - at least once: alespoň jednou
 - at most once: nejvýše jednou
 - exactly once: právě jednou
 - no guarantees: bez záruky o počtu retransmisí

Ztracená odpověď

- Rozlišení typu požadavků
 - Idempotentní: je možno zopakovat (bez vedlejších efektů)
 - * Ztráta řešena prostou retransmisí
 - Neidempotentní: má vedlejší efekty
 - * Retransmise musí být zpracována serverem
 - Rozezná retransmisi
 - Neopakuje tělo
 - Vydá výsledek, pouze pokud jej má v cache

Zhroucení serveru

- Pořadí požadavek versus výpadek
- Výpadek před zpracováním požadavku
 - Retransmise
- Výpadek před odesláním odpovědi
 - Vráť chybu
- Client není schopen zjistit příčinu

Zhroucení klienta

- Procedura zůstává na serveru
 - zmařený výkon CPU
 - drží zdroje (např. zamčené soubory)
 - může odpovídat „podruhé“ po rebootu klienta
- Možné řešení: *soft služby* (nikoliv součást RPC)

RPC – literatura

- Douglas E. Comer: Computer Networks and Internets, Prentice Hall, 2001 (3. vydání); kapitola 33
- SUN developers guide: <http://docs.sun.com/?q=ONC+RPC>
- Přednášky:
<http://www.cee.hw.ac.uk/courses/5nm1/index.htm>

Síťové systémy souborů

- Použití RPC pro přístup k vzdálenému systému souborů
- Network File System – NFS
 - Nedefinuje systém souborů, ale *vzdálený přístup* k němu
 - Nezávislý na operačním systému, systému souborů i architektuře
 - * Původně vyvinut firmou Sun
 - * Cílem nabídnout přístup i mimo svět UNIXu
 - Typický model klient/server
 - * Lokální požadavky na operace systému souborů jsou posílány na vzdálený server
 - * Pro tento účel se využívá RPC

Připojení NFS

- Připojení (mount) prvním krokem
 - Stejný princip jako u lokálních systémů souborů
 - Částečná závislost na OS (adresář vs. označení „disku“)
- Server drží seznam oprávnění (ACL) pro klienty
- Připojení mohou být kaskádována

NFS Protokol

- NFS protokol definuje operace, které musí server podporovat
 - Čtení a zápis souboru
 - Přístup k atributům souboru
 - Vyhledání souboru v adresáři
 - Čtení a práce s adresáři
 - Práce s odkazy (links)
- Všechny tyto operace jsou implementovány jako RPC
 - Zpravidla prostřednictvím démona (nfsd)
 - Lokální operace transformována do RPC volání serveru
 - Server implementuje tato volání nad svým lokálním systémem souborů

Bezstavovost

- NFS nemá operace otevření a zavření souboru
- NFS je *bezstavové*
 - Standardní server si nepamatuje požadavky ani klienty
 - Každá RPC musí poskytnout všechnu potřebnou informaci
- Výhody
 - Robustní, v principu neovlivněno výpadky serveru
- Nevýhody
 - Zápisy musí být synchronní, klient očekává, že data byla skutečně zapsána

Konzistence

- Pouze v omezeném rozsahu
 - Předpokládá se, že konzistenci si klienti zajistí nezávisle na NFS
- Atomicita operací
 - Write má být atomický, ale nemusí se vejít do jediného paketu
 - * Může pak dojít k prokládaným RPC

Konzistence II

- Vyrovnávací paměti
 - Server
 - * Pro čtení bez problému, zápis ne
 - Klient
 - * Problematická, vyrovnávací paměť na klientu (1) není přístupná klientu (2)
 - * Problematické pro čtení i zápis

Distribuované objekty

- DO – distribuce enkapsulovaných dat na počítačích v síti
 - Pouze data (funkce nejsou přenositelné) – distribuovaná enkapsulovaná data
 - Skutečně přenositelné funkce (Java)
- Složené dokumenty
 - Data plus prohlížeč či editor
 - Původně pouze pro výměnu dat mezi jednotlivými aplikacemi
 - Později přidána „distribuce“
- *Distribuované objekty nebo distribuované komponenty (data plus funkce)*

Distribuované objekty II

■ Objekt

- Enkapsulace dat: **stav**
- Operace nad daty: **metody**
- Zpřístupnění metod: **rozhraní**

■ Distribuce

- Jednoduchá
 - * Server: Data a metody
 - * Klient: Rozhraní

Vzdálené objekty

- Plně distribuované
 - * Data a metody distribuovány přes několik serverů
 - * Klient: Rozhraní

Vazba klienta

- Proxy
 - Implementace rozhraní objektu
- Implicitní vazba
 - Přímé volání vzdálené metody
 - Klient nerozezná lokální a vzdálený objekt
- Explicitní vazba
 - Speciální funkce pro navázání objektu
 - Teprve pak jsou zpřístupněny jeho metody

Odkazy na objekty

- Charakter globálního ukazatele
- Základní informace
 - Adresa stroje s objektem
 - Identifikace objektu
- Location server
 - Nalezení objektu
 - Registrace na globální server
 - * Potenciální úzké místo
- Nezávislost přenosových protokolů
 - Výběr vhodné proxy

Plná abstrakce

- Skrytí všech závislostí
- Odkaz je tvořen *implementační „ručkou“* (Implementation handle)
 - Kompletní implementace proxy
 - Skrývá veškeré detaily (místo, protokol, . . .)
- Bezpečnostní implikace

Statické a dynamické volání

- **Statické volání (invocation)**
 - Predefinované definice rozhraní
 - Analogie RPC
 - Buď IDL nebo interní (Java)
- **Dynamické volání (invocation)**
 - Výběr metody proveden až během výpočtu
invoke(object, method, input_param, output_param)
 - Příklad:
string.append(data)
invoke(string, id(append), data)
id(append) je funkce vracející identifikátor metody

Implementace rozhraní

- Aplikace má klientskou a serverovou část
- Klient komunikuje prostřednictvím *stubs*
- Server komunikuje prostřednictvím *skeletonů*

Stubs

- Lokální reprezentace vzdáleného objektu
 - Proxy
- Zajišťují
 - Inicializaci a volání vzdáleného objektu (analogie `clnt_create`)
 - Zpracování argumentů (XDR)
 - Informace o začátku volání (`clnt_call`)
 - Převzetí výsledků, případně výjimek
 - Ukončení volání (`clnt_destroy`)

Skeleton

- Zanoření metody v serveru
- Výběr konkrétní implementace metody
- Zajišťují
 - Převzetí argumentů
 - Vlastní provedení/volání metody
 - Předání argumentů

Registry

- Adresář
- Server registruje metodu
 - Asociuje jméno se vzdáleným objektem
- Klient hledá vzdálenou metodu jménem
 - Metoda je volána poté, co je její záznam nalezen v Registry

Objektově orientované přístupy

- Příklady realizace
 - JAVA RMI
 - CORBA
 - DCOM

Java RMI

- Java Remote Method Invocation
 - Plnohodnotný objektový přístup
- Pouze vzdálené objekty
- Malé rozdíly mezi lokálními a vzdálenými objekty
 - Klonování
 - * Pouze na serveru
 - * Explicitní (nová) vazba
 - Synchronizace
 - * Nad vzdálenými objekty není úplná (pouze na proxy)

Vyvolání objektu

- Opět pouze malé rozdíly mezi lokálním a vzdáleným voláním
- Serializace parametrů
 - Vlastnost, umožňující enkapsulaci a předání objektu (parametru)
 - * Platformově závislé objekty (např. deskriptor souboru či soket) nejsou serializovatelné
- Lokální objekty předávány hodnotou
- Vzdálené objekty předávány referencí

Odkazy

- Vzdálený objekt je definován dvěma třídami
 - Server class: implementace kódu na straně serveru
 - Client class: implementace proxy
- Proxy je serializovatelná
 - Je možné předávat odkaz na vzdálený objekt
 - Teoreticky celý byte kód proxy

Java RMI

- **Shrnutí vlastností**

- Neomezené vyvolání objektů na vzdálených strojích
- Plná integrace distribuovaných objektů
- Podpora zpětných volání (callback) ze serveru
- Bezpečnost a spolehlivost

- **Pokročilé možnosti**

- Možnost aktivace persistentních objektů
- Garbage collection i na distribuovaných (vzdálených) objektech
- Replikace serverů

DCOM

- Distributed Components Object Model firmy Microsoft
 - Postaven na COM
 - * Compound documents
 - * OLE (Object Linking and Embedding)
 - * Definován ve 300 stránkové specifikaci (1995)
 - DCOM nabízí komponenty na vzdálených strojích
 - * Maximalizace transparentního přístupu (access transparency)

DCOM

- Základní komponenty architektury
 - Microsoft IDL (MIDL)
 - Interface Identifier (IID), 128 bitů
 - Objekty typu třída (a CreateInstance)

CORBA

- Common Object Request Broker
- Standard, definovaný OMG (Object Management Group)
<http://www.omg.org>
- Typický reprezentant modelu klient-server
- Základní cíl:
 - Zajistit, aby objekty distribuované v síti dokázaly spolupracovat.
- Základní funkce
 - Nalezení objektu
 - Směrování požadavku
 - Vyvolání metody
 - Navrácení výsledku

Object Management Architecture

- **Systemové komponenty**
 - Object Request Brokers (ORBs)
 - Object Services
- **Aplikační komponenty**
 - Společné služby
 - Aplikační objekty

Object Request Brokers

- Propojují jednotlivé komponenty (objekty)
- Zajišťují vyhledávání
- Zprostředkovávají posílání zpráv

Objektové služby

- Vytváření a odstranění objektů
- Relokace a replikace objektů
- Hlídní přístupu k objektům

Interface Description Language

- V podstatě zajišťuje přemostění mezi implementacemi
- Definiuje rozhraní client/server

Úložiště

- **Interface Repository**
 - Uchovává rozhraní k objektům (alternativa k IDL)
- **Implementation Repository**
 - Informace o umístění objektu a prostředí (OS, ...)
 - Informace o implementaci (verze, ...) – vhodné pro ladění
 - Použita při aktivaci objektu
 - Implementace se musí registrovat

Služby

- Základní služby v CORBA 3.0 (první část)

Added základní architektura pro workflow systémy

Structuring

Mechanisms for

OTS

Collection vytváření a manipulace se skupinami objektů

Concurrency podpora souběžných procesů, používá transakční služby a zamykání

Enhanced View reprezentace hodin (časovačů)

of Time

Events práce s událostmi (registrace, publikace a přijímání událostí)

Externalisation export/import objektů

Služby – pokračování

- Základní služby v CORBA 3.0 (druhá část)

<i>Licensing</i>	licence, licenční politiky
<i>Life Cycle</i>	vytváření, kopírování, přesun a destrukce objektů
<i>Management of Event Domains</i>	práce s obory událostí
<i>Naming</i>	jmenná služba (pro objekty)
<i>Notification</i>	rozšíření základní práce s událostmi
<i>Persistent State</i>	persistentní objekty a manipulace s nimi
<i>Query</i>	databázové dotazy

Služby – pokračování

- Základní služby v CORBA 3.0 (třetí část)

Relationship explicitní práce se vztahy mezi objekty a vlastnostmi

Security bezpečnost, autentizace, autorizace, šifrování

Telecoms Log logování událostí

Time podpora operací s časem a časovými známkami

Trading Object v podstatě vyhledávací služba (pro služby)

Transaction transakce (včetně dvoufázového commitu)